NAVAL POSTGRADUATE SCHOOL
Monterey, California

Using Local Optimality Criteria for Efficient Information Retrieval
with Redundant Information Filters

by Neil C. Rowe[1]

March 1994

Approved for public release; distribution is unlimited.

Prepared for:

DARPA
3701 N. Fairfax Drive
Arlington, VA 22203-1714

Naval Postgraduate School
Monterey, CA 93943

NAVAL POSTGRADUATE SCHOOL
Monterey, California

REAR ADMIRAL T. A. MERCER
Superintendent

HARRISON SHULL
Provost

This report was prepared by:

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	

4. PERFORMING ORGANIZATION REPORT NUMBER(S) NPSCS-94-004	5. MONITORING ORGANIZATION REPORT NUMBER(S) Naval Postgraduate School

6a. NAME OF PERFORMING ORGANIZATION Computer Science Dept. Naval Postgraduate School	6b. OFFICE SYMBOL (if applicable) CS	7a. NAME OF MONITORING ORGANIZATION ARPA
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943		7b. ADDRESS (City, State, and ZIP Code) 3701 N. Fairfax Drive Arlington, VA 22203-1714

8a. NAME OF FUNDING/SPONSORING ORGANIZATION Naval Postgraduate School	8b. OFFICE SYMBOL (if applicable) NPS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER AO 8939

8c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943	10. SOURCE OF FUNDING NUMBERS			
	PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Using Local Optimality Criteria for Efficient Information Retrieval with Redundant Information Filters

12. PERSONAL AUTHOR(S)
Neil C. Rowe

13a. TYPE OF REPORT Progress	13b. TIME COVERED FROM Jan 93 TO Dec 93	14. DATE OF REPORT (Year, Month, Day) 1994 3	15. PAGE COUNT 35

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	filters, optimization, queries, conjunction, boolean algebra, natural language

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

We consider information retrieval when the data, for instance multimedia, is conputationally expensive to fetch. Our approach uses "information filters" to considerably narrow the universe of possibilities before retrieval. Then decisions must be made about the necessity, order, and concurrent processing of proposed filters (an "execution plan"). We develop simple polynomial-time local criteria for optimal execution plans, and show that most forms of concurrency are suboptimal with information filters. Although the general problem of finding an optimal execution plan is likely exponential in the numbers of filters, we show experimentally that our local optimality criteria, used in a polynomial-time algorithm, nearly always find the global optimum with 15 filters or less, sufficient number of filters for most applications. Our methods do not require special hardware and avoid the high processor idleness that is characteristic of massived-parallelism solutions to this problem. We apply our ideas to an important application, information retrieval of captioned data using natural-language understanding, a problem for which the natural-language processing can be the bottleneck if not implemented well.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT. [ ] DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Neil C. Rowe	22b. TELEPHONE (Include Area Code) (408) 656-2462	22c. OFFICE SYMBOL CSRp

DD FORM 1473, 84 MAR   83 APR edition may be used until exhausted   SECURITY CLASSIFICATION OF THIS PAGE
All other editions are obsolete

Using Local Optimality Criteria for Efficient Information Retrieval with Redundant Information Filters

Neil C. Rowe[1]

Code CS/Rp, Department of Computer Science
Naval Postgraduate School
Monterey, CA USA 93943
rowe@cs.nps.navy.mil

ABSTRACT

We consider information retrieval when the data, for instance multimedia, is computationally expensive to fetch. Our approach uses "information filters" to considerably narrow the universe of possibilities before retrieval. Then decisions must be made about the necessity, order, and concurrent processing of proposed filters (an "execution plan"). We develop simple polynomial-time local criteria for optimal execution plans, and show that most forms of concurrency are suboptimal with information filters. Although the general problem of finding an optimal execution plan is likely exponential in the number of filters, we show experimentally that our local optimality criteria, used in a polynomial-time algorithm, nearly always find the global optimum with 15 filters or less, a sufficient number of filters for most applications. Our methods do not require special hardware and avoid the high processor idleness that is characteristic of massive-parallelism solutions to this problem. We apply our ideas to an important application, information retrieval of captioned data using natural-language understanding, a problem for which the natural-language processing can be the bottleneck if not implemented well.

1. Introduction

Information retrieval of multimedia data differs from traditional information retrieval in that the data can be so much costlier to fetch because it is so much larger. Thus high recall (retrieval of all relevant data in the database) and high precision (yield of relevant data in the fetched set) are more important than in citation retrieval. So more effort is needed before data fetch, and it is important to find the best way to do it. The best ways are not necessarily the same as the best ways for traditional database systems, as discussed in work such as [3, 5, 12, 19].

We are exploring the concept of "information filters" [2] to improve query performance. These processes take as input a set of media-object pointers, and return the subset that pass some necessary but not sufficient conditions for a data match. Different filters can work on separate parts of a query, or on separate media if each datum is multimedia (as when pictures have associated text captions or audio). We assume here that filters err only on the side of caution, so that they never exclude relevant media objects. Even though detailed examination of the data would subsume their results, information filters can be cost-effective if their cost is significantly less than a data match. But not all filters are cost-effective, nor all ways of using them, and we need to develop a theory for using them.

Signature matching [4, 8, 10, 11] is a special case of information filtering that has been fruitfully applied first to text data and then to multimedia data. It extracts the key words in text, the key shapes in pictures, or the key sounds in audio, and hashes them into a "signature table". At query time, query words or features are also hashed into the signature table. A hash hit on any word or feature is a necessary but not sufficient condition for an exact match between the query and some multimedia datum that was hashed there. The signature file can be stored in main memory, and searching it can be considerably faster than searching a secondary-storage index to the data. Thus signature matching is a special case of information filtering as we defined it above. Signature matching does usually require, however, that each media object be described manually, the description must anticipate most future queries (which is difficult, as discussed in [20]), and the match be exact.

Some ways of signature matching are better than others. As an example, suppose a user wants to find a picture

of an F-18 aircraft on a runway in a database of captioned pictures, a typical query of those we have been studying for the Photo Lab of the Naval Air Warfare Center, Weapons Division (NAWC-WD), China Lake, California, USA [21]. This could be decomposed into four filters applied to the picture database: (1) pictures of F-18s, (2) pictures of runways, (3) pictures of things "on" other things, and (4) pictures of F-18s on runways. Intuitively, it would seem best to apply the first three filters in that order because "F-18" is quite specific, "runway" less so, and "on" even less so. Intuitively, it would also seem that the third filter could be eliminated in deference in the fourth, because almost always an aircraft is on a runway and not beside it, underneath it, or in some other relationship to it. But we need mathematics to justify these intuitions.

The primary objective of this paper is to present a general theory of optimal information retrieval with information filters. While the theory applies to all kinds of filters, it contains important new results about signature and other redundant filters, which have not previously been carefully analyzed as abstracted components of an information-retrieval system. We will use a simple model of filter cost that corresponds closely to that of most information-retrieval implementations. We will first examine in sections 2 and 3 the most common kind of multi-filtering, conjunctive filtering, and provide simple local optimality conditions on a conjunctive sequence. The local optimality conditions concern interchanges of filter order, deletion of redundant filters, insertion of redundant filters, and concurrent execution of filters. Surprisingly, we will prove that with a general cost model, most forms of concurrency are not desirable with information filtering, since the earlier starts of the concurrent filters do not compensate for the increased input they must handle. Section 3 will show results of experiments confirming the value of our local optimality criteria, and in particular that a simple "greedy" algorithm based on them has excellent average performance.

We then generalize our results to arbitrary boolean expressions involving filters in section 4. Disjunctive sequences are just the duals of conjunctive sequences, and negations are relatively straightforward in their optimality implications. Factoring of conjuncts over disjuncts and vice versa leads to an additional local optimality condition, but one that we argue is unimportant in most applications.

One important application of signature matching is to supporting natural-language processing in information retrieval. Many researchers in information retrieval have pointed out the deficiencies of raw keyword matching

(e.g. [16]), and parsing and semantic interpretation of natural-language data descriptions could be a solution. The main obstacle is speed. Depending on the approach, the required parsing, semantic interpretation, and semantic matching could take minutes where keyword matching requires seconds. But if we can decompose the required processing into several filters, we may be able to rule out most potential data at an early stage, without full natural-language processing of their data descriptions. We discuss this in section 5 of this paper, and our theory permits us to improve the MARIE system [21], which pioneered in the systematic use of natural-language captions on multimedia data as the primary indexing of the data.

2. General analysis of conjunctive information filtering

Suppose we have m information filters through which some data items must pass; that is, each data item must pass the test administered by each filter. Let the event of passing the filter be termed f_i. Assume each filter has an average cost of execution per data item of c_i and an average a priori probability of passing the data item of $p(f_i)$, where $0 < p(f_i) < 1$ to avoid considering trivial cases. Generally the c_i will be execution times so as to find the minimum execution time of a filter sequence, but our mathematics here applies to any costs. Assume that the cost of testing whether an item passes a filter is a constant independent of the success or failure of the test; this is true of table lookups, for instance.

If the filters are applied in sequence, the expected total cost per data item will be:

$$t_{1,m} = c_1 + c_2 p(f_1) + c_3 p(f_1 \wedge f_2) + \cdots + c_m p(f_1 \wedge f_2 \cdots \wedge f_{m-1}) \tag{1}$$

We would like to choose the filter sequence that minimizes $t_{i,m}$ for a set of m filters, or know if possible deletions of filters could improve cost. The parameters c and p can be estimated either from past statistics of the filters on similar problems, or by random sampling of a small fraction of the database and applying the filters to it.

Related problems to the finding the optimal conjunctive sequences have been examined elsewhere. In the database literature this is the problem of restriction-order optimization based on selectivities (called a "single-variable" optimization in [15]), but the focus there has been on solving the more critical problem of optimizing joins and other operations that generally are far worse bottlenecks in processing time for databases. The usual

methods require search, either exhaustive or heuristic, in the space of possible rearrangements of a query expression, rather than attempting to find general optimization criteria. Work in semantic query optimization for database queries sometimes suggests signature-table methods [23], though it is usually concerned with application of more complicated "integrity constraints". In artificial intelligence, [25] analyzed optimization of filter sequences that create persistent variable bindings, a different problem but related to the one above. Work in Markovian decision processes [17] has developed general methods for situations more complicated than sequences, but these are not very efficient for sequences. Work on optimal decision trees assumes all c_i terms are equal, which leads to specialized algorithms. Problems of task scheduling that are related to conjunctive-sequence ordering are generally NP-complete [9] because generally we must to examine some constant fraction of all possible sequences in order to find the optimal one. But in this paper, we will propose some quick polynomial-time criteria that can be used to rule out all but a few possible sequences, as for instance criteria that sort the sequence. While these criteria are not guaranteed to find the optimal solution, they usually do, as we have confirmed by experiments, and furthermore they usually greatly improve the average-case execution time of the sequence.

2.1. Local criteria for interchange-optimality of the cost of a conjunctive filter sequence

First, consider the effect on cost of interchanging filters in a conjunctive sequence. If a sequence is a local optimum with respect to cost, then any such interchange must not decrease the total cost. Consider the effect of interchanging adjacent filters i and $i+1$. This certainly cannot affect the cost terms for terms before i, and it cannot affect the cost terms for terms after $i+1$ because $f_1 \wedge f_2 = f_2 \wedge f_1$. So the interchange of filters i and $i+1$ will not improve cost if:

$$c_i p(f_1 \wedge f_2 \wedge \cdots f_{i-1}) + c_{i+1} p(f_1 \wedge f_2 \wedge \cdots f_{i-1} \wedge f_i) \le c_{i+1} p(f_1 \wedge f_2 \wedge \cdots f_{i-1}) + c_i p(f_1 \wedge f_2 \wedge \cdots f_{i-1} \wedge f_{i+1})$$

or, after converting to conditional probabilities which represent the "value" of each filter:

$$c_i + c_{i+1} p(f_i | f_1 \wedge f_2 \wedge \cdots f_{i-1}) \le c_{i+1} c_i p(f_{i+1} | f_1 \wedge f_2 \wedge \cdots f_{i-1})$$

we then get after rearranging:

$$c_i / [1 - p(f_i | f_1 \wedge f_2 \wedge \cdots f_{i-1})] \le c_{i+1} / [1 - p(f_{i+1} | f_1 \wedge f_2 \wedge \cdots f_{i-1})] \tag{2}$$

In other words, a locally optimal sequence must be sorted by c/q, where q is the fraction of the items failing this filter after passing all previous filters. We call this "interchange optimality".

Often we can assume that filter-acceptance events f_i and f_{i+1} are independent of all the previous filter-acceptance events. Then the conditional probabilities become the a priori probabilities $p(f_i)$ of a random data item passing a filter i, and we get (Theorem 1 of [14]):

$$c_i/(1-p(f_i)) \le c_{i+1}/(1-p(f_{i+1})) \tag{3}$$

2.2. Entailing and entailed filters

Unfortunately, we cannot often assume independence of filter pairs because the whole idea of signature matching is that anything that passes full matching will also pass the signature matching. Hence the conditional probability for the signature filter passing a data item that the full-matching filter passes is 1. We will call the signature filter the "entailed" filter, and the full-matching filter the "entailing" filter. We will assume that entailment, or conditional dependence of the probability of filter success, is always absolute if it occurs at all. Filters usually can be designed to accomplish this, though it should be noted that if f_3 entails f_1 and f_3 entails f_2, then f_1 and f_2 cannot be completely independent, although they could very near independence.

With an entailing filter e, $p(f_e|u) \ne p(f_e)$ where u represents the situation of passing all the previous filters before e. But we can use Bayes' Rule. Suppose u can be broken into two pieces such that $u = u_1 \wedge u_2$ and u_2 is the largest subset independent of f_e. Then:

$$p(f_e|u) = p(f_e|u_1) = p(f_e)/p(u_1) \tag{4}$$

To obtain $p(u_1)$ if all the filters in u_1 are independent, multiply their probabilities. Otherwise, since entailment is absolute when it does occur, eliminate the probabilities of filters that are entailed by others in u_1 and then multiply the rest.

For instance, if filter 7 entails filters 3 and 4 but is independent of four other preceding filters, and the a-priori probability of passing filter 3 is 0.4, of passing filter 4 is 0.5, of passing filter 7 is 0.1, and filter 3 is near-independent of filter 4, $p(f_7|f_1 \cdots \wedge f_6) = 0.1/(0.4*0.5) = 0.5$. If filter 7 is then followed by a filter 8 of the same cost but with an independent success probability of 0.6, filters 7 and 8 should not be interchanged in search of local optimality.

Another consequence of our assumption of all-or-none dependence between filters is that $p(f_i|u)$ is only different from $p(f_i)$ when it is an entailing filter, and then it is only a function of its entailed filters. But entailed filters must precede all their entailing filters in a given filter sequence to be useful. So $p(f_i|u)$ will be a constant for all sensible placements of an entailing filter f_i in a filter sequence. Hence inequalities (2) and (3) are sorting criteria for a bubble sort on the filter sequence. Hence if we bubble-sort using interchange optimality, and the resulting sequence also obeys entailment relationships, we have found the optimal order for the sequence in polynomial time with respect to the number of filters. If the sort result does not obey entailment relationships, we must try something else. Section 3.1 discusses what else we can do, and gives an important theorem for this situation.

Note that even if entailment is not all-or-none, result (2) can still allow sorting of a filter sequence if the variations in the conditional probabilities $p(f_i|f_1 \cdots \wedge f_{i-1})$ are sufficiently small with to assign a consistent place to any f_i in the sequence.

2.3. Deleting entailed filters

Entailed filters must come before their entailing filters or else they are useless. There is now a new possibility for improving filter-sequence cost without changing the answers it produces: deletion of an entailed filter.

Assume first that there is only one entailing filter and only one entailed filter. Assume the entailed (fast) filter is i and the entailing (slow) filter is e. As with interchange, the deletion cannot affect the cost terms before i. It cannot either affect the cost terms after e because filter e will rule out an item regardless of how much assistance it gets from filter i. So deletion of filter i does not improve cost if:

$$c_i p(f_1 \wedge \cdots f_{i-1}) + c_{i+1} p(f_1 \wedge \cdots f_{i-1} \wedge f_i) + c_{i+2} p(f_1 \wedge \cdots f_{i-1} \wedge f_i \wedge f_{i+1}) + \cdots + c_e p(f_1 \wedge \cdots f_{e-1})$$
$$\leq c_{i+1} p(f_1 \wedge \cdots f_{i-1}) + c_{i+2} p(f_1 \wedge \cdots f_{i-1} \wedge f_{i+1}) + \cdots + c_e p(f_1 \wedge \cdots f_{i-1} \wedge f_{i+1} \cdots f_{e-1})$$

or after introducing conditional probabilities:

$$c_i + c_{i+1} p(f_i|f_1 \cdots \wedge f_{i-1}) + c_{i+2} p(f_i \wedge f_{i+1}|f_1 \cdots \wedge f_{i-1}) + \cdots + c_e p(f_i \cdots \wedge f_{e-1}|f_1 \wedge \cdots f_{i-1}) \qquad (5)$$
$$\leq c_{i+1} + c_{i+2} p(f_{i+1}|f_1 \wedge \cdots f_{i-1}) + \cdots + c_e p(f_{i+1} \cdots \wedge f_{e-1}|f_1 \wedge \cdots f_{i-1})$$

We call this condition "deletion optimality".

If we can assume that passing of filter f_i is independent of passing all other filters f_j from $j=1$ to $e-1$ we get after rearrangement:

$$c_i/(1-p(f_i)) \le c_{i+1}+c_{i+2}p(f_{i+1}|f_1 \wedge \cdots f_{i-1})+ \cdots +c_e p(f_{i+1} \cdots \wedge f_{e-1}|f_1 \wedge \cdots f_{i-1}) \qquad (6)$$

Note that the right side is just $t_{i+1,e}$ in the notation of equation (1), or the expected cost of the filter sequence $f_{i+1}, f_{i+2}, ..., f_e$ by itself.

Another way to simplify (5) is to note that $p((f_i \wedge r)|f_1 \wedge \cdots f_{i-1}) \le p(r|f_1 \wedge \cdots f_{i-1})$, and we can use this to match each pair of the last $e-i-1$ terms on the left side and the right side. Then eliminating each pair, we get a simpler sufficient condition for (5) to be true:

$$c_i/[1-p(f_i|f_1 \wedge \cdots f_{i-1})] \le c_{i+1} \qquad (7)$$

Note that condition (7) for filter i implies the interchange optimality condition (3) for filters i and $i+1$, since $c_{i+1}<c_{i+1}/(1-p(f_{i+1}))$.

2.4. Deletion of more than one entailed filter

A question arises about deletion optimality: Even if filters i and j are individually deletion-optimal in a particular filter sequence, could the deletion of *both* of the them be locally optimal? The following definitions and Lemma will provide the criteria for such stronger forms of optimality.

--A filter that is not entailed is "strongly-deletion-optimal". An entailed filter i is strongly-deletion-optimal if condition (7) holds, $c_i/(1-p(f_i|f_1 \cdots \wedge f_{i-1})) < c_{i+1}$, and if at the same time $c_{i+1} \le c_j$ for all $i<j\le r$, where r is the next non-entailed filter after i if any, else the last filter.

--A filter that is not entailing is "strongly-interchange-optimal" if it is interchange-optimal by (2) with respect to all other filters. An entailing filter i is strongly-interchange-optimal if $c_j/(1-p(f_j|f_1 \cdots \wedge f_{j-1})) < c_i/(1-p(f_i))$, for all $r \le j<i$ where f_j is not entailed by f_i and where r is the last non-entailed filter before i.

Lemma 2.1, Subsequence Deletion Suboptimality Lemma: Given a set of filters in which every filter pair is either probabilistically independent or else one filter in the pair entails the other. Suppose filter i in some sequence S is strongly deletion optimal. Then strong deletion optimality of the filter originally at position i must also hold

for any subsequence created by deleting items besides i of S. Proof: If filters deleted from sequence S occur after $i+1$, they do not affect either side of (7), so it still holds. If filters deleted are independent of filter i, they cannot affect either side of (7). If a filter j before filter i is entailed by i (i could not entail j for the sequence to make sense), and j is deleted, the conditional probability can only decrease since $p(f_i|b) < p(f_i|b \wedge c)$, so (7) still holds. The only remaining case is when filter $i+1$ is deleted, and some filter j to the right of it now follows i in the sequence. Then if $c_{i+1} < c_j$ for any filter j that could become the next filter after i by deletions, then (7) holds for the new filter. We only need to consider filters up to the next nonentailed filter, because that one cannot be deleted. QED.

We can use this lemma to get sufficient conditions to say that a filter sequence is the globally optimal one with respect to interchanges and deletions. The conditions require only polynomial time to confirm. This result applies to an important class of problems, and filters can be purposely designed to make global optimality easier to guarantee.

Theorem 2.1, Restricted Global-Optimality Theorem: Given a set of filters in which every filter pair is either probabilistically independent or else one filter in the pair entails the other. Assume in some filter sequence S that every filter is strong-interchange-optimal, and every entailed filter is strong-deletion-optimal. Then S is the global optimum in the space of improper subsequences created by deletions from it and permutations of it. Proof: By Lemma 1, any subsequence T created solely by deletions (with no interchanges) must also be strong-deletion-optimal. Since each T can be created by a single deletion from another strong-deletion-optimal sequence S, no T can be the global optimum because it is more costly than its S.

But sequence permutation could improve the cost. For this, we need only consider the moving left of an entailing filter E because (a) entailing filters are the only ones whose interchange optimality is affected by deletions, and (b) their ratio $c/(1-p)$ will be decreased by deletions of leftward filters, so they may need to be moved left to put the sequence back into sorted order on $c/(1-p)$. However, if strong-interchange-optimality holds, interchange optimality cannot hold with the new left neighbor of E no matter where E is moved. Furthermore, if strong-interchange-optimality holds with all filters left of E through the first non-entailed filter, deletions of any of them cannot affect the strong-interchange-optimality of the remaining filters because $p(f_i) \leq p(f_i|f_k)$. QED.

Note two important cases to which the Theorem applies: if there are no entailed filters in a sequence, or if one entailing filter entails all the others (as in "multi-level" signature matching [4]).

The conditions of the Theorem require only $O(m^2)$ operations to perform, m the number of filters, while at the same time pruning a possibly exponential number of possible permuted subsequences. So a good heuristic for finding an optimal sequence for a set of filters is to first interchange-sort the filters, then check if the conditions of Theorem 1 hold; if they do, you have the global optimum and are done. Theorem 1 can also rule out from consideration subsequences of the original filter sequence even when it does not apply to the original sequence.

2.5. Inserting entailed filters

Another way to improve the cost of a sequence is to insert a filter redundant with respect to an existing filter, before that latter filter. (It makes no sense to insert a nonredundant filter, as without such a filter, the final output of the filter sequence will be different.) This is the exact opposite of the case in the last section, so we just reverse the direction of the inequalities for local optimality, with the proposed insertion filter as the filter i. A way to rule out insertion-optimality is to find another sequence consisting of the given sequence plus one extra filter, where the extra filter is deletion-optimal. This is straightforward to accomplish if filter sequences are considered in order of decreasing size.

2.6. Distributed filtering

So far we have only considered sequential implementation of a conjunctive filter sequence. A distributed implementation could assign each filter to a processor, send each filter processor each data item, and have all the processors send the data that passes their tests to a single "intersection processor" that finds items that passed more than a threshold number of filters. There are two problems with this approach: the intersection processor is a bottleneck, and the degree of parallelism in the first phase is limited by the number of filters which can be devised. More sophisticated information-retrieval systems today allow boolean expressions with embedded conditions instead of keyword lists. A similar distributed approach can also be followed but with a more complicated final aggregation step, and hence an even worse bottleneck.

Another approach is to assign to sets of data items to processors, which is possible on a massively parallel computer. [27] explored this idea on the Connection Machine, a massively parallel machine with 64,000 processors, with each processor independently comparing keywords with its own associated item or items. The keywords in sequence were supplied to all processors simultaneously, and each processors counted the number of matches for each data item. At the conclusion of this, processors were polled to get the answers. This approach can get answers very fast, and the speedup should be almost linear with the number of processors used. But this massive parallelism also means massive idleness: Usually most data items do not match the query, and their processors just sit uselessly. So this approach is very wasteful of computer resources, something which is hard to justify for a non-critical application like information retrieval and a multi-million dollar machine like the Connection Machine. Instead, we will explore partition of the data items with a significantly lower degree of parallelism, an approach that could work for networked workstations.

2.7. Data-partition parallelism

Suppose we have N processors that can help execute the filters, where each processor filters a randomly chosen disjoint partition of the input set. Each would first apply filter 1 to its partition, then filter 2 to the output of filter 1, etc. Assume that the cost of applying a filter to a set of data items is proportional to the number of data items; this is true for most filters, including signature table methods as well as the natural-language processing filters discussed later in this paper. Then without overhead, each filter will execute N times less time. Otherwise, we can assume that overhead cost is proportional (with constant k_o) to the number of processors used. So the total cost of doing a sequence of filters with data-partition parallelism is $k_o N + (c_1/N) + (p(f_1)c_2/N) + \cdots$, which has a minimum with respect to N at:

$$N_{best} = \sqrt{(1/k_o)(c_1 + p(f_1)c_2 + \cdots)} \qquad (6)$$

This must be rounded to an integer, so this true optimum is usually only approached.

Usually this is a large number, larger than N, since k_o represents the time to send simple messages to other processors, which is usually much faster than executing filters themselves. Then since the derivative of the cost is negative for $N < N_{best}$, the minimum occurs at N, and it best to use all N processors. This is only different for

filters that are all very simple, with hardware that poorly supports message passing, or situations with a large number of available processors (something difficult to justify for information retrieval). And with a sequence of filters, the setup cost $k_o N$ need only be incurred once because each processor applies the data to each filter in turn, so its cost can be amortized; and the final union of results is just a disjoint union, easy to accomplish.

But we need not assign all the processors to the same filter at the same time. We could assign some of the N processors to one filter and the remainder to another filter. Surprisingly, we can prove this is never desirable, under a few simple assumptions.

Theorem 2.2, Parallel-filter Theorem: Suppose we have N processors to implement two information filters. Suppose the cost, per unit number of data items, of n processors doing a filter i is $g(n)+(c_i/n)$, $g(n)$ a communications-cost function (covering the processor setup and result-list intersection for the parallel processing), and c_i the cost of the filter per data item as above. That is, we decompose cost into a linear-speedup term and an overhead-cost term. Assume $g''(n) \leq 0$ and $g(0)=0$ (concavity would reflect economies of scale in invoking processors). Then it is best to apply all N processors to one filter, then all N processors to the other filter.

Proof: Suppose we do assign n_1 processors to filter 1 and $N-n_1$ processors to filter 2, where $n_1 \leq N$. Then execution time for just the two filters plus overhead will be $g(n_1)+g(N-n_1)+\max((c_1/n_1),(c_2/(N-n_1)))$. As a function of real n_1, (c_1/n_1) is monotonically decreasing and $(c_2/(N-n_1))$ is monotonically increasing. Hence the minimum of the max term will be when $c_1/n_1=c_2/(N-n_1)$, or when $n_1=Nc_1/(c_1+c_2)$, at which value the cost attains a minimum of $g(Nc_1/(c_1+c_2))+g(Nc_2/(c_1+c_2))+((c_1+c_2)/N)$. On the other hand, if all N processors are assigned to perform filtering operation 1, then filtering operation 2, the cost of the two filters plus overhead will be $g(N)+(c_1/N)+(p_1c_2/N)$ where p_1 is the probability of passing filter 1. If we ignore the overhead terms, the parallel-filter (first) approach cannot be preferable since that would mean $(c_2/N)<(p_1c_2/N)$, which is impossible since $p_1<1$. Considering just the overhead terms, the parallel-filter approach's g terms would only be preferable if $g(n)+g(N-n)<g(N)$ for some integer n, $0 \leq n \leq N$. But this is impossible because $g''(n) \leq 0$, so $g(N/2) \geq (g(0)+g(N))/2=0.5g(N)$ and $g(n)+g(N-n) \leq g(0)+g(N)$. Hence we have shown that both filter cost and overhead cost are inferior for the parallel-filter approach.

The above analysis is actually conservative. Note that to implement parallel filtering, we must round from

$Nc_1/(c_1+c_2)$ to an integer; this can only worsen the cost further. Furthermore, the above results assume that with sequential filtering, all N processors do filter 1, then all N do filter 2, etc. But if, say, processor 6 finishes filter 1 early, it could start applying filter 2 to its results of filter 1, before processor 5 finishes filter 1. This interleaving effect is data-dependent and hard to analyze, but could only be better without parallel filtering, because then each processor does each filter and there are more opportunities to finish early and do the next filter. QED.

The following corollary generalizes this result on two filters to arbitrary execution plans involving parallelism for a set of filters.

Corollary 2.1: Given an execution plan for a set of filters on N processors, expressed as a directed acyclic graph with each node marked as to which filter and which of N subdivisions of the data to which it applies, and where a filter can appear more than once. Suppose the cost, per unit number of data items, of n processors doing a filter i is $g(n)+(c_i/n)$, $g(n)$ a communications-cost function (covering the processor setup and result-list intersection for the parallel processing), and c_i the cost of the filter per data item as above; and assume $g''(n) \leq 0$ and $g(0)=0$. Then if that execution plan has different filters in parallel anywhere, it is not optimal. Proof: Such an execution plan could be transformed into a sequence of filters by repeatedly taking a pair of parallel strands and sequencing them arbitrarily. Then if we reverse the order of these transformations, we will get the original execution sequence. But in this latter process, Theorem 2 applies at every step, so the original execution plan cannot be optimal even if the completely sequenced plan is optimal. Furthermore, if a filter appeared more than once in the original execution plan, it will appear more than once in the completely sequenced plan, so that plan cannot be optimal anyway. QED.

As an example of the Corollary, suppose we have three filters f_1, f_2, and f_3. Then putting f_1 in parallel with the sequence of f_2 followed by f_3 cannot be optimal because the last two can be considered a composite filter. Similarly, suppose we do f_1 then f_3 on one parallel track, f_2 then f_3 on another, where the starting time of f_3 on the first track could be different than the starting time on the second; then sequencing would have each filter once plus f_3 twice, which is worse than just having each filter once.

2.8. Parallel non-filtering processes

Information-filtering applications can require additional non-filtering processing. In natural-language information retrieval, for instance, parsing and interpretation of the natural language is necessary before detailed matching can be done. The effect of such processes can be modeled as imposing earliest start times for all the processes that depend on them. This introduces additional inequality constraints of a more traditional sort into our scheduling problem.

We can handle these constraints in the standard manner of optimization theory. If the local optimality conditions can be satisfied without violating the new start-time constraints, the local optimum remains a local optimum. Otherwise, the local optima must be on the border of the region of feasibility with the minimum number of "active" constraints (inequalities reducing to equalities). That means that any local optimum of the new problem must satisfy interchange optimality and local deletion optimality except in the minimum number of places necessary to satisfy the start-time constraint. Standard algorithms can solve such problems.

2.9. General algorithm for sequence optimization

We can now provide a general method for filter-sequence optimization. First, sort the filters for interchange optimality. If Theorem 2.1 applies to this sequence, and it obeys dependencies, it is the global optimum. Otherwise, rearrange to satisfy the dependencies, and conduct a branch-and-bound search: Consider possible subsequences of the sorted filter sequence created by deleting entailed filters that are not strongly-deletion-optimal and resorting to preserve interchange optimality. Any such subsequence that satisfies Theorem 2.1 and dependencies represents a local optimum for the original sequence of filters, and furthermore none of its subsequences can be a local optimum. Then if N processors are available, assign all N for each filter in turn if $N < N_{best}$, otherwise use N_{best}. This algorithm has exponential time complexity in the number of filters because of the possibly exponential number of combinations that must be considered, but it is simple and works.

3. Experiments with random data

Analysis of filter execution plans can vary greatly in difficulty depending on the parameters of the filters involved. To better judge the number of local optima and how often the global optimum is easy to find, we conducted experiments with randomly generated filters. Given a particular number of filters to create, we randomly designated certain ones as entailing filters, and randomly chose some filters for them to entail. Entailment relationships were restricted to form a forest, since signature-table filters do, and it is difficult and not very useful for filter f_A to be entailed by both filters f_B and f_C when f_B and f_C are unrelated. Note that deletion of a node from a forest and rerouting the children nodes maintains the forest property. (The forest restriction does allow a filter f_D to entail both f_B and f_C; for instance, f_D could be a full match to the data item, f_B a match to its high-order bits, and f_C a match to its low-order bits.)

Costs were randomly assigned to each filter from the uniform distribution 0 to 10. Filter success probabilities were assigned from the uniform distribution 0.01 to 0.99; for nonentailing filters, this was taken as the a priori probability, and for entailing filters, this was taken as the conditional probability given that all previous filters succeeded.

Fig. 1 shows an example run with four randomly generated filters. The first argument to "filter" is the filter number, the second its average cost, and the third its probability information. "Dep" means the filter entails those listed, so filter 2 entails 1, and filter 3 entails both 1 and 2. There are four sequences of filter subsets that satisfy interchange optimality and the dependencies. Of these, two satisfy deletion optimality criteria (more often, only one does). The first is the true optimum, and the heuristic greedy algorithm finds it. Note how deletion interacts with interchange optimality: Filter 4 normally should precede filter 3, but when both filters 1 and 2 are deleted, filter 3 becomes more valuable and must precede 4.

Fig. 2 tabulates experimental results which are graphically represented in Figs. 3-11. Each row of Fig. 2 represents 1000 randomly generated filter sets. The first column is the number of filters, the second the probability that a filter would be selected for possible entailment (although it would be ruled out if such an entailment would violate the forest property), and the third the probability that a filter is entailing (that is, whether we

should attempt to construct entailment relationships for it). The remaining columns show experimental results as means of natural logarithms, with associated standard errors in parentheses. We use the logarithms because they are more appropriate for summarizing combinatorial experiments. The fourth column is the mean of the logarithms of the number of possible subsequences that need to be considered, after interchange sorting, for each set of randomly generated filters. The fifth column is the mean of the logarithms of the number of those subsequences that were judged locally optimal with respect to the criteria of section 2, a number generally considerably smaller than that of the previous column. Note the values in the fifth column increase more slowly than the values in the fourth column.

Fig. 3 plots the size of the search space, the total number of sequences $n!$ for n filters, with the values displayed in fourth column of Fig. 2; four assignments of dependency parameters are shown. Figs. 4-7 show these last four curves plotted against the number of locally optimal sequences, the fifth column of Fig. 2.

3.1. A greedy algorithm

The sixth column of Fig. 2 shows the performance of a simple heuristic "greedy" algorithm to find the optimum, in terms of the means of the logarithms of the ratios of the cost of the sequence found by the algorithm to the cost of the true optimum sequence. This algorithm starts with the interchange-sorted (see discussion below) list of filters, and successively deletes the best filter that it can (that is, the entailed filter whose deletion improves overall cost the most after resorting), until no further deletion can improve cost. No backtracking is done. This heuristic algorithm is $O(m^3)$, m the number of filters, since interchange-sorting is $O(m \log m)$, and there are $O(m)$ things to delete and hence $O(m)$ steps; each step looks at $O(m)$ subsequences and evaluates the cost of each subsequence in $O(m)$ time, then resorts in $O(m)$ time to reposition one entailing filter. The heuristic algorithm cannot get the optimal solution in every case of the general problem. But the sixth column demonstrates that it nearly always gets the correct answer for up to fifteen filters, and its rate of deterioration is considerably slower than the increases in the size of the problem space, the number of sequences considered, and the number of local optima. Figs. 8-11 plot columns 5 and 6 of Fig. 2 against one another.

The greedy algorithm is supported by the following useful theorem that relates interchange optimality and

deletion optimality. This says that we need only interchange-sort the filter set in order to check for a local optimum: If the interchange sorting does not obey entailment relationships, it is not locally optimal.

Theorem 3.1: If a conjunctive filter sequence is sorted so as to satisfy interchange optimality, using for each entailing filter its conditional probability given all entailed filters are present, and this sequence violates entailment relationships, the sequence is not deletion-optimal. Proof: Suppose a sequence is interchange-sorted so that entailing filter f_e occurs before its entailed filter f_d, where d is the costliest in $c/(1-p)$ of all such entailed filters after e. Then $c_e/(1-p(f_e|u_e))<c_d/(1-p(f_d|u_d))$. Now consider moving d before e in order to satisfy entailment. Since d is the ratio-costliest of filters entailed by e and after e, it is also the ratio-costliest of all filters entailed by e, and must go just before e to achieve the locally-optimal ordering of all sequences with d before e. But with d just before e, the condition for deletability of d is that $c_d/(1-p(f_d|u_d))>c_e$. But this follows from the original assumptions since $c_e<c_e/(1-p(f_e|u_e))$. QED.

4. Disjunctions and negations

We now extend the previous analysis to filter execution plans that are equivalent to arbitrary boolean expressions.

4.1. Ordering of disjunctions

We expect that disjunctions of filters will be rather rare, as conjunctions of conditions are more useful in information retrieval. But in general, optimization of disjunctions is precisely analogous to (the "dual" of) optimization of conjunctions. The following theorem generalizes Theorem 2 of [14] beyond independent filters.

Theorem 4.1: The problem of optimally ordering a disjunctive sequence of filters is equivalent to optimally ordering a conjunctive sequence in which the costs are the same and the probabilities are mapped to their inverses. Proof: If the filters are applied sequentially, then everything that fails the first filter is applied to the second filter, and everything that fails both filters is applied to the third filter, and so on. The final answer is just the appending of results from all filters, since this union is disjoint. Hence the cost formula is

$c_1+c_2p\,(\tilde{}f_1)+c_3p\,(\tilde{}f_1\tilde{\wedge}f_2)+\cdots$, identical to that for conjunctive sequences except with the inverse probabilities. But the inverse function $f(x)=1-x$ is monotonic and has the same domain and range. So the problem of finding an optimal disjunctive sequence has an exact "dual" problem of finding the optimal conjunctive sequence for the inverse of the original probabilities. The solution to the latter found by the abovementioned methods then maps to the solution for the former. QED.

Note that this also provides a criterion for deletion of redundant disjunctive filters, and implies that concurrent execution of different filters in disjunctions is also not advantageous.

4.2. The distributive laws

If a filter expression includes both conjunctions and disjunctions, should we factor it (using the distributive laws) to improve execution time? Surprisingly, the answer is an unequivocal "yes."

Theorem 4.2: With three arbitrary nontrivial filters, the execution plan $f_1\wedge(f_2\vee f_3)$ is preferable to the equivalent plan $(f_1\wedge f_2)\vee(f_1\wedge f_3)$. Proof: Let u represent all events before f_1 is evaluated. The first execution plan is better than the second if:

$$c_1 + c_2p\,(f_1|u) + c_3p\,(f_1\tilde{\wedge}f_2|u) < c_1 + c_2p\,(f_1|u) + c_1p\,(\tilde{}f_1\tilde{\vee}f_2|u) + c_3p\,(f_1\tilde{\wedge}f_2|u)$$

which simplifies to $0<c_1p\,(\tilde{}f_1\tilde{\vee}f_2|u)$, which is always true for nontrivial filters. QED.

Hence an additional local optimality condition for an execution plan involving both conjunctions and disjunctions is that all possible factorings be made for conjunctions over disjunctions. A similar result holds for disjunctions over conjunctions.

Theorem 4.3: The execution plan $f_1\vee(f_2\wedge f_3)$ is always preferable to the equivalent plan $(f_1\vee f_2)\wedge(f_1\vee f_3)$. Proof: Similarly to the preceding, we get:

$$c_1 + c_2p\,(\tilde{}f_1|u) + c_3p\,(\tilde{}f_1\tilde{\wedge}f_2|u) < [c_1 + c_2p\,(\tilde{}f_1|u)] + [c_1p\,(f_1\vee f_2|u) + c_3p\,(\tilde{}f_1\tilde{\wedge}f_2|u)]$$

And all terms cancel except for the third on the right side, $c_1p\,(f_1\vee f_2|u)$. QED.

Note that Theorems 4.2 and 4.3 do not require probabilistic independence. And the f_i terms can be composite

(or logical combinations of other filters), so the result applies to the distribution of conjunction over more than two disjunctions, and vice versa. But the above results do only provide local optimality conditions, because some boolean expressions can be factored in more than one way. For instance, there are two local optima for:

$$(a \wedge b) \vee (a \wedge c) \vee (b \wedge c) = (a \wedge (b \vee)) \vee (b \wedge c) = (a \wedge b) \vee ((a \vee b) \wedge c)$$

However, this possibility can usually be safely ignored, since conjunctions are the most natural way to conjoin nearly all filters; and when disjunctions do occur, it is unlikely that a filter must appear twice in any locally-optimal execution plan, as is necessary above.

4.3. Redundancy elimination in boolean expressions

A special case of the distributive law is:

$$f_1 \vee (f_1 \wedge f_2) = (f_1 \wedge true) \vee (f_1 \wedge f_2) = f_1 \wedge (true \vee f_2) = f_1$$

The final expression must execute faster than the original expression because it is a subexpression. The other "absorption" law is $f_1 \wedge (f_1 \vee f_2) = f_1$.

In general, all such redundancy-elimination laws of logic can be fruitfully applied to boolean expressions of filter combinations. They include:

$$f \wedge f = f, \ f \vee f = f, \ f \wedge true = f, \ f \wedge false = false, \ f \vee true = true, \ f \vee false = f$$

All these involve substitution of an expression requiring less work to evaluate.

4.4. Negations

Negation operators will complete a boolean algebra of filter expressions. Since filters always operate on a finite set, it is helpful to think of the negations as being set differences on either the results of filters previously passed, if any, or otherwise the full database. And we will automatically eliminate double negations from a boolean expression.

We will assume that the negation of a noncomposite filter has the same evaluation cost as the unnegated filter. This is true for signature tables and other filters wherein a similar calculation is performed upon every input data

item, and the result used to decide if the item passes the test; more complex filters that do not fulfill this restriction can often be decomposed into boolean combinations of subfilters that do. Under this assumption, we can prove that negations in a boolean filter expression should be pushed as far as possible inside expressions, so that they all apply to single filters and thus can be evaluated at no cost penalty.

Theorem 4.4: Consider an equivalence class of boolean expressions such that any member of the class can be transformed into any other member by some sequence of applications of DeMorgan's Laws. Then if this expression is interpreted as referring to information filters, the globally optimal member of the class is that in which every negation is of a noncomposite (simple) filter. Proof: There must be only one such expression, because the laws above apply independently to each negation sign. Any other expression in the equivalence class must be derivable by a series of applications of the "negation-factoring" forms of DeMorgan's Laws.

First consider $f_1 \wedge f_2$ versus $\tilde{\ }(\tilde{\ }f_1 \vee \tilde{\ }f_2)$. If u represents the previous context of the subexpressions, and f_1 and f_2 are simple filters, the first expression costs $c_1 + p(f_1|u)c_2$, and the second costs $c_1 + p(f_1|u)c_2 + c_n$, where c_n is the "negation cost", the cost per data item of checking which items in a set do *not* belong to u. Hence with simple but nontrivial filters, the first form is always better because all the terms are the same except for the added negation-cost term in the second expression. Second, consider $f_1 \vee f_2$ versus $\tilde{\ }(\tilde{\ }f_1 \wedge \tilde{\ }f_2)$. The first expression costs $c_1 + p(\tilde{\ }f_1|u)c_2$, and the second costs $c_1 + p(\tilde{\ }f_1|u)c_2 + c_n$. Again, the second cost is worse for simple nontrivial filters.

If f_1 and f_2 are composite in the above filter expressions, however, the transformation from the first form to the second could decrease cost if f_1 and f_2 are themselves negated expressions, in which case double negations cancel. However, the cost calculation for the second form of each pair always adds at least one c_n term. Hence its cost is always worse than that of an expression in the same equivalence class with no c_n terms in its cost, the expression with negations pushed all the way inward. QED.

Besides DeMorgan's, a few other laws of logic that can help simplify an expression. Certainly $f_1 \wedge \tilde{\ }f_1 = false$ and $f_1 \vee \tilde{\ }f_1 = true$ are desirable substitutions. But the "negative absorption" laws $f_1 \wedge (\tilde{\ }f_1 \vee f_2) = f_1 \wedge f_2$ and $f_1 \vee (\tilde{\ }f_1 \wedge f_2) = f_1 \vee f_2$ are unnecessary since they just represent the way we execute filters.

4.5. Simulated annealing for finding optima

Fig. 12 summarizes the main results of this paper so far. We have covered enough of the laws of boolean logic, the propositional calculus, to generate all equivalent expressions to an arbitrary input expression, since we have covered all those listed in [15]. The middle column shows that the first four classes of logical equivalences do require some cost and probability analysis; but the methods of section 3 will do this, and they are not hard. The rightmost column shows that three of the seven classes of equivalences are possible problems for a "greedy" algorithm which sequentially applies the best equivalence until it reaches a local optimum. These three are what make the general problem difficult and probably exponential in complexity. However, results of section 3 suggests that a polynomial-time greedy algorithm can get the right answer most of the time. If this is insufficient assurance of optimality, simulated annealing can explore the search space randomly until some given level of assurance is achieved. The necessary random transformations of the boolean expression can include all the methods in Fig. 12.

5. Experiments with a natural-language processing-filter application

We now discuss a specific filtering application to which we have applied our theory. This application illustrates a number of subtleties in the use of filters. It also is valuable in its own right, as one of the easiest ways for users to access multimedia databases. The idea is provide information retrieval of multimedia data with natural-language (in our case, English) questions as input. Examples of this approach are [13] and [22] and the more complicated ideas reviewed in [23]. Natural-language processing poses a good challenge for filtering ideas because it can require much time, yet it is not so slow as to fail to benefit from a modest improvement in efficiency.

We wish to improve our earlier MARIE system [21] with the ideas of this paper. The goal of MARIE is to take as input a English noun phrase representing a query, and return as output the multimedia data items that match the meaning (as opposed to literal words) of the query, doing most processing with the pointers to those data items and not the items themselves. The domain of MARIE is captioned photographs in the Photo Lab of NAWC-WD, China Lake, California. The conceptual units of the improved MARIE will be:

1. *Coarse-grain matcher (C)*: a keyword match of nouns in the English query input to caption nouns, using index files [21]. It returns a set of pointers to media data items. This process must use a type hierarchy to be truly helpful, as observed in [6] and [26].

2. *Parser (P)*: a natural-language understanding system that parses English query input and creates a *meaning list*, its logical form [21]. We assume input and captions exhibit "conjunctive semantics" [1] where the meaning of the whole is the conjunction of a set of logical expressions that define the meaning of the parts, a usually reasonable assumption.

3. *Registration-data tester (R)*: a formatted-condition processor, like those in database query languages, that returns pointers to data items matching registration-data (formatted non-caption) conditions in the query input. Registration data at NAWC-WD includes date, location, photographer, type of film, and security classification. This tester was implemented for this paper using a main-memory database.

4. *Picture-type matcher (T)*: a process that identifies the possible broad classes to which a media datum or a query can belong (like "test" or "historical" or "public relations" for photographs), and rules out media data whose classes are incompatible with the query classes [20].

5. *Fine-grain matcher (F)*: a graph matcher that checks whether the query input graph (representing the query meaning list) is isomorphic to some part of some caption graph (representing the caption's meaning list) [21]. Like the coarse-grain matcher, this needs a type hierarchy, and it also needs a part-whole hierarchy. It helps to separate this from the coarse-grain match, as did [7, 18], since it requires combinatorial analysis and can be much slower.

Fig. 13 shows the dependencies between the processes. Each of these can be a separate process on a separate processor; input and output queues can enable asynchronous communication. Items 1, 3, 4, and 5 are conjunctive information filters as we defined them in section 2; and item 2 imposes a minimum start-time constraint on 4 and 5. Two additional filters that could also be added are an automatic "content-analysis" filter that analyzes the media data for things and relationships displayed within, and the human user who accepts or rejects what the computer eventually supplies. We could also add separate filters for additional textual, audio, and video aspects

of a datum, if these could be analyzed separately.

5.1. Mathematical analysis of the four MARIE filters

We can exhaustively consider all possible orderings. Entailments can be summarized:

--C-R: independent, following argument above

--C-T: approximately independent (two different kinds of reasoning)

--C-F: first entailed by second

--R-T: independent (they examine different sorts of data)

--R-F: independent

--T-F: first entailed by second

Then the only possible filter sequences obeying dependencies are:

4-filter: CRTF, CTRF, RCTF, RTCF, TCRF, TRCF, CTFR, TCFR

3-filter: RTF, CRF, TRF, RCF, TFR, CFR

2-filter: RF, FR

We randomly chose 230 captions from the NAWC-WD Photo Lab database. We used 44 test queries, 42 supplied by the Photo Lab personnel as typical of the queries they receive everyday, and two longer queries from [20]. We obtained average CPU times per data item (in seconds) and average success probabilities in experiments with an implementation in Quintus Prolog on a Sun SparcStation. These parameters, plus the ratios $r_i = c_i/(1-p_i)$, are:

$c_C = 0.0102$, $p_C = 0.0305$, $r_C = 0.0105$

$c_R = 0.000602$, $p_R = 0.958$, $r_R = 0.0144$

$c_T = 0.000236$, $p_T = 0.749$, $r_T = 0.000939$

$c_F = 3.11$, $p_F = 0.421$ *(conditional on C and T)*, $r_F = 5.37$

Note how a redundant filter can still be highly useful: Coarse-grain rules out an average of 97% of the database in very little time.

If we could ignore the parser (as when certain queries are common, and their meaning lists stored in advance),

the interchange sort of these four filters would be TCRF (picture-type matcher, coarse-grain matcher, registration-data tester, and fine-grain matcher). This order satisfies the dependencies. It would also satisfy deletion optimality since $0.000939 < 0.0102$ for deletion of T and $0.105 < 0.00602 + (0.958 * 3.11)$ for deletion of C. C would be strong-deletion-optimal since it is not followed by an entailed (deletable) filter; T would be strong-deletion-optimal because $0.000236 < 0.0102$. F would satisfy strong interchange-optimality because $0.000602 < 5.37$. Hence TCRF would be globally optimal.

But the parser imposes an minimum time for T and F from the start of processing. The parser is going to be on the critical path for the optimal execution plan, because even if C and R are done sequentially while the parser is executing, $230 * (0.0102 + (0.0305 * 0.00602)) = 2.39$ is the expected time for the sequence C-R, and we obtained an average of $c_P = 3.76$ seconds for parse CPU time for the 44 queries. Hence the parser and F will be on the critical path (F cannot be deleted), and must occur in that order; if T occurs, it must be between the parser and F because of the dependencies. T must occur because it is strongly-deletion-optimal, since $0.000236 < 3.11$. Hence the critical path is parser-T-F.

As for parallelism, the parser cannot be decomposed into parallel tasks as currently implemented. And parallelism is of no advantage to C and R because they are not on the critical path, even if they are taken sequentially. So execute sequentially C-R on a processor to run in parallel with the parser. As for T and F, we must estimate the overhead of parallelism. We conducted experiments with the Quintus Prolog communications package TCP using 1, 2, 3, and 4 processors for the fine-grain matcher. In these experiments, we observed no statistical effect of the number of processors on overhead cost. So we fit cost to the formula $k_o + (c_F / N)$ for the cost of distributing fine-grain over N processors, and estimated $k_o = 0.4$ as the amortized overhead per data item. Hence T and F should be allocated the maximum number of available processors each, and all of T's executions must precede all of F's executions, following Theorem 2.2. Fig. 12 summarizes the optimal execution plan.

To confirm the reasonability of the cost estimates, we conducted further tests. Among other things, we measured real time to the nearest second for execution of the four filter sequences CF, CTF, F, and TF on a Sun SparcStation. We measured real time to be sure to include all the factors that affect execution time, but the workstation used a fileserver that serves other users, so our measurements are not precise. In 42 queries of the

previously supplied 44 (two of which showed new bugs and were excluded), the observed ratio of real execution time for F to TF was 1.18 with a standard deviation of 0.43, versus a theoretical ratio of 1.33; and the ratio of F to CF was 22.1 with a standard deviation of 17.3, versus a theoretical ratio of 29.7. The observed ratio of real execution time for CF to CTF was 2.20 with a standard deviation of 1.50, versus a theoretical ratio by our above methods of 1.33; the data was in the form of small integers averaging about 20, so this measure was more crude. In these same experiments, F was never faster than CF and TF was never faster than CTF, as predicted by theory; and F was faster than TF as predicted in only 9 out of 42 cases, and only slightly faster in each of the 9. These results are confirmation of our theory considering that our method of parameter estimation, by averaging over all queries, biases performance toward the few queries with many answers.

5.2. Scaling up the database

The MARIE system was just a prototype implementation that handled a random sample of 1/166 of the entire NAWC database, which at the time of the sample was 36,000 captioned data items. We can use the preceding analysis to predict the optimal execution plan when MARIE is applied to the full database.

The total time to do fine-grain matching and picture-type matching should increase by 166 since each match is independent and there are no economies of scale. Registration-data testing will be close to 166 times slower because it is best implemented with indexes, and the indexes will be 166 times longer on the average. Coarse-grain matching will be dominated by the intersection of lists 166 times longer on the average, so it will also be close to 166 times slower. Only the parser will remain nearly the same speed, since the grammar it handles is nearly a complete grammar for the remaining captions, and most of the parse time is in fetch from a hashed lexicon, backtracking among grammar choices, and reasoning about lexicon information for words, activities not affected by the size of the database.

That means the effective time cost for the parser is reduced to 3.76/36000=0.000104 per data item. This still rules out the sequence TCRF which was optimal without considering the parser, but still allows the sequences CRTF, CTRF, RCTF, RTCF, CTFR, RTF, CRF, RCF, CFR, RF, and FR. Deletion optimality is not affected by the inclusion of the parser, so we need only consider the four-filter sequences CRTF, CTRF, RCTF, RTCF, and

CTFR. The optimal solution when a single inequality constraint is imposed upon a problem is one in which the inequality constraint is "active" or at the border of infeasibility, which means T must be second in the sequence, leaving only CTRF, RTCF, and CTFR as possibilities. But in the second of those R precedes C, and in the third of those F precedes R, both of which are inconsistent with interchange optimality. Hence sequence CTRF is the optimal one, with the parser in parallel with C. As before, there is no benefit to putting any of the filters in parallel with different filters, but there is an advantage in data-partition parallelism on each filter. So the optimal execution plan with N available filters is to put 1 processor on the parser in parallel with N-1 processors on C, then N processors on the sequence TRF on different random partitions of the database.

5.3. Other modifications of MARIE

Our approach also permits straightforward analysis of several interesting hypothetical modifications. If the parser finds natural-language input to be ambiguous, alternative meaning lists can be generated. Then the T and F processors can execute a disjunction of the alternatives, using the methods of section 4.

Another idea is to split the coarse-grain filter into separate filters for each noun of the query. The current implementation uses a single heap structure for all input nouns, but a simpler implementation would load and intersect index files, finding captions belonging to each of a set of index files (assuming an exact match to the query is desired). Then large index files have both a high cost and a high probability of success, hence a high ratio $c/(1-p)$. Hence if we partition the coarse-grain matching into separate filters for each noun, the filters should be sorting by increasing frequency of noun occurrence. This may mean that other filters like R and T can now be interleaved with coarse-grain filters, and placed before filters for high-frequency nouns (like "view" and "test" at NAWC-WD).

Yet another idea is to treat the user as another filter, as we suggested earlier. The user will examine data items supplied, and will accept some of them. We can assume that any user knows better what they want than any automatic filter, so this user "filter" U entails all the others discussed; but to maintain a tree structure for entailments, U must only directly entail fine-grain matching (F). Since F must go last without this user filter, U when present must be last. The only question then is deletion optimality of the filters. The deletion of C or T cannot

affect U because they are "buffered" by F. The deletion of R is pragmatically impossible to affect U because $r_R = 0.0144$ in the current MARIE implementation, which is far less than the time it would take a human to assess the relevance of a picture. The deletion of F is only desirable if $5.37/N > c_U$, N the number of processors. Thus if time of the user to confirm answers is the only cost criterion, we should only delete F if the user averages less than $5.37/N$ seconds to assess a picture; if other criteria were included in the cost like bother to the user, this threshold would decrease. If F should be deleted, then we can next consider deleting other filters, but their values of the ratio r are so much lower that it is pragmatically impossible for this to be desirable.

5.4. Further capabilities with a mixed query language

Beyond the capabilities just described, we have implemented for MARIE an enhanced query capability in a SQL-like format, allowing arbitrary nesting of boolean expressions, including possibly multiple natural-language strings and multiple registration-data restrictions. The methods of sections 2 and 4 can be applied to these enhanced queries. Our query language adds an additional comparator "MATCHES" to SQL, to initiate natural-language processing and semantic matching. It does not handle joins, however, since we believe that good captions and a good type hierarchy can eliminate most need for them in multimedia databases. The availability of such a "mixed" query language with both conventional SQL and natural-language-descriptor features means that the natural-language processing does not need to handle complicated scoping rules for quantifiers like "not", "or", and "all", which can be very tricky to analyze in English, since the user can express such distinctions with the formal part of the query language. Programmers can use our modified SQL directly, but we also provide a graphical interface for naive users that permits structured query formulation.

6. Conclusions

We have explored a new approach to information retrieval, the concept of information filtering. Our approach has focussed on the system aspects of filtering rather than the details of the filters, and our work should nicely complement the results on filter design provided by classic information-retrieval methods and work on signature-based retrieval. Our approach has been mostly analytical, providing local optimality criteria for filter

execution plans. Thus it contrasts with work on query optimization for database systems, for which the search space is so difficult to analyze that methods must be either exhaustive or heuristic; our local optimality criteria are considerably more sophisticated than the "cheapest-first" heuristic often used there. Information filtering thus appears to be a special case of general database retrieval that has special exploitable properties for improving efficiency. The methods we have proposed will be particularly useful for the design of multimedia information-retrieval systems, for which conceptually distinct filters can easily be derived.

7. References

[1] Allen, J. *Natural language understanding*. Menlo Park, CA: Benjamin Cummings, 1987.

[2] Belkin, N. J. and Croft, W. B. Information filtering and information retrieval: two sides of the same coin? *Communications of the ACM, 35*, 12 (December 1992), 29-38.

[3] Bertino, E., Rabitti, F., and Gibbs, S. Query processing in a multimedia document system. *ACM Transactions on Office Information Systems, 6*, 1 (January 1988), 1-41.

[4] Chang, J. W., Hyuk, J.C., Sang, H. O., and Lee, Y. J. Hybrid access method: an extended two-level signature file approach. International Conference on Multimedia Information Systems, ACM, Singapore (1991), 51-62.

[5] Chang, S. K., Yan, C. W., Dimitroff, D. C., Arndt, T. An intelligent image database system. *IEEE Transactions on Software Engineering, 14,* 5 (May 1988), 681-688.

[6] Chen, H., Lunch, K. J., Basu, K., and Ng, D. T. Generating, integrating, and activating thesauri for concept-based document retrieval. *IEEE Expert, 8*, 2 (April 1993), 25-34.

[7] Cohen, P. R. and Kjeldsen, R. Information retrieval by constrained spreading activation in semantic networks. *Information Processing and Management, 23*, 4 (1987), 255-268.

[8] Constantopoulos, P., Drakopoulos, J., and Yeorgaroudakis, Y. Retrieval of multimedia documents by pictorial content: a prototype system. International Conference on Multimedia Information Systems, ACM, Singapore (1991), 35-48.

[9] El-Rewini, H., Lewis, T., and Ali, H. H. *Task scheduling in parallel and distributed systems*. Englewood Cliffs, NJ: Prentice-Hall, 1994.

[10] Faloutsos, C. Signature-based text retrieval methods: a survey. *Database Engineering*, March 1990, 27-34.

[11] Faloutsos, C. and Christodoulakis, S. Description and performance analysis of signature file methods for office files. *ACM Transactions on Office Automation Systems, 5*, 3 (July 1987), 237-257.

[12] Gibbs, S., Tsichritzis, D., Fitas, A., Konstantas, D., and Yeorgoroudakis, Y. (1987). Muse: A multimedia filing system. *IEEE Software, 4*, (2), 4-15.

[13] Grosz, B., Appelt, D., Martin, P. and Pereira, F. TEAM: An experiment in the design of transportable natural language interfaces. *Artificial Intelligence, 32* (1987), 173-243.

[14] Hanani, M. Z. An optimal evaluation of boolean expressions in an online query system. *Communications of the ACM, 20,* 5 (May 1977), 344-347.

[15] Jarke, M. and Koch, J. Query optimization in database systems. *Computing Surveys, 16,* 2 (June 1984), 117-157.

[16] Krovez, R. and Croft, W. B. Lexical ambiguity and information retrieval. *ACM Transactions on Information Systems, 10,* 2 (April 1992), 115-141.

[17] Mine, H. and Osaki, S. *Markovian decision processes.* New York: American Elsevier, 1970.

[18] Rau, L. Knowledge organization and access in a conceptual information system. *Information Processing and Management, 23,* 4 (1988), 269-284.

[19] Roussopoulous, N., Faloutsos, C., and Sellis, T. An efficient pictorial database system for PSQL. *IEEE Transactions on Software Engineering, 14,* 5 (May 1988), 639-650.

[20] Rowe, N. Inferring depictions in natural-language captions for efficient access to picture data. To appear in *Information Processing and Management,* 1994.

[21] Rowe, N. and Guglielmo, E. Exploiting captions in retrieval of multimedia data. *Information Processing and Management, 29,* 4 (1993), 453-461.

[22] Sembok, T. and van Rijsbergen, C. SILOL: A simple logical-linguistic document retrieval system. *Information Processing and Management, 26,* 1 (1990), 111-134.

[23] Shekhar, S., Srivastava, J., and Dutta, S. A formal model of trade-off between optimization and execution costs in semantic query optimization. *Data and Knowledge Engineering, 8* (1992), 131-151.

[24] Smeaton, A. F. Progress in the application of natural language processing to information retrieval tasks. *The Computer Journal, 35,* 3 (1992), 268-278.

[25] Smith, D. and Genesereth, M. Ordering conjunctive queries. *Artificial Intelligence, 26,* (1985), 171-215.

[26] Smith, P., Shute, S., Galdes, D., and Chignell, M. Knowledge-based search tactics for an intelligent intermediary system. *ACM Transactions on Information Systems, 7,* 3 (July 1989), 246-270.

[27] Stanfill, C. and Kahle, B. Parallel free-text search on the Connection Machine system. *Communications of the Assocation for Computing Machinery, 29,* 12 (December 1986), 1229-1239.

filter(1,3.79283,[0.207059,indep]).
filter(2,2.18817,[0.0621229,dep,1]).
filter(3,8.6706,[0.0347425,dep,2,1]).
filter(4,8.63665,[0.0574234,indep]).

Cost-prob ratios: [4.78325,2.33311,8.98268,9.16281]

[1,2,4,3]: 4.81338
[2,4,3]: 2.75564 (local optimum)
[3,4]: 8.97066 (local optimum)
[1,4,3]: 5.68422

The number of local optima: 2
Heuristic optimum: [2,4,3]

Figure 1: Example output of the conjunctive-sequence tester

no. of filters	prob. entailed	prob. entailing	size of space	number of optima	heuristic cost ratio
3	0.2	0.2	0.0554518(0.0188046)	0.0138629(0.00970406)	0.0(0.0)
4	0.2	0.2	0.152492(0.0303405)	0.00693147(0.00689673)	0.0(0.0)
5	0.2	0.2	0.263396(0.0446852)	0.038712(0.0172595)	0.0(0.0)
6	0.2	0.2	0.505997(0.0641714)	0.0762462(0.0216879)	0.0(0.0)
7	0.2	0.2	0.526792(0.0651116)	0.112081(0.0272095)	0.0(0.0)
8	0.2	0.2	0.672353(0.0749465)	0.114958(0.0266312)	0.0(0.0)
9	0.2	0.2	0.977338(0.0930445)	0.156547(0.0322006)	0.0(0.0)
10	0.2	0.2	1.21301(0.113208)	0.168711(0.0340448)	0.0(0.0)
11	0.2	0.2	1.2338(0.116638)	0.238026(0.039367)	0.0(0.0)
12	0.2	0.2	1.45561(0.133149)	0.235557(0.0417515)	0.0145623(0.0144893)
13	0.2	0.2	2.07251(0.145889)	0.343708(0.0472216)	0.0(0.0)
14	0.2	0.2	2.25273(0.147813)	0.35064(0.0516756)	0.0(0.0)
15	0.2	0.2	2.12796(0.161744)	0.327977(0.054881)	0.0125881(0.00800236)
16	0.2	0.2	2.74486(0.170891)	0.471194(0.0614881)	0.0151046(0.0142948)
17	0.2	0.2	3.04985(0.179684)	0.478125(0.0579669)	0.0008991(0.0006283)
18	0.2	0.2	3.39642(0.185603)	0.622168(0.0740848)	0.0008681(0.0008096)
19	0.2	0.2	4.07571(0.212578)	0.832629(0.0828184)	0.0521339(0.0289764)
20	0.2	0.2	4.19354(0.199061)	0.695475(0.0637804)	0.0360905(0.0264117)
3	0.2	0.8	0.277259(0.0449211)	0.00693147(0.00689673)	0.0(0.0)
4	0.2	0.8	0.547586(0.0557068)	0.0207944(0.0118242)	0.0(0.0)
5	0.2	0.8	0.998132(0.065259)	0.103972(0.0247503)	0.0(0.0)
6	0.2	0.8	1.26846(0.0832094)	0.128821(0.0294552)	0.0(0.0)
7	0.2	0.8	1.85763(0.0887012)	0.253066(0.0391569)	0.0120575(0.00854089)
8	0.2	0.8	2.38443(0.10481)	0.360936(0.0487895)	0.0130564(0.012991)
9	0.2	0.8	2.80725(0.0986974)	0.46442(0.0569815)	0.0332153(0.0252187)
10	0.2	0.8	3.16768(0.10009)	0.550636(0.056419)	0.0347772(0.0245527)
11	0.2	0.8	4.08957(0.108937)	0.619519(0.0599677)	0.0317728(0.0316135)
12	0.2	0.8	4.42921(0.113004)	0.708135(0.0682391)	0.018403(0.016857)
13	0.2	0.8	5.15702(0.114451)	0.664393(0.068758)	0.0346816(0.0187861)
14	0.2	0.8	5.85709(0.100626)	0.78386(0.0816895)	0.0176791(0.0119071)
15	0.2	0.8	6.40468(0.1131)	0.89792(0.0758427)	0.0402969(0.0223391)

Figure 2, page 1

| no. of filters	prob. entailed	prob. entailing	size of space	number of optima	heuristic cost ratio
3	0.8	0.2	0.270327(0.448086)	0.0277259(0.0135829)	0.0(0.0)
4	0.8	0.2	0.60997(0.681121)	0.0277259(0.0135829)	0.0(0.0)
5	0.8	0.2	0.89416(0.971866)	0.0415888(0.0164613)	0.0(0.0)
6	0.8	0.2	1.44175(0.119125)	0.0855333(0.0264176)	0.0(0.0)
7	0.8	0.2	1.6081(0.144361)	0.157725(0.0355865)	0.0(0.0)
8	0.8	0.2	2.18341(0.170456)	0.122422(0.0320054)	0.0135923(0.0132327)
9	0.8	0.2	3.02212(0.174673)	0.156026(0.0367928)	0.00639268(0.00636064)
10	0.8	0.2	3.37563(0.190245)	0.194092(0.0383726)	0.00763135(0.00753294)
11	0.8	0.2	3.81924(0.219739)	0.262072(0.0491324)	0.0782128(0.0489747)
12	0.8	0.2	4.92134(0.222132)	0.251086(0.0458398)	0.049436(0.0312112)
13	0.8	0.2	5.55211(0.233519)	0.229114(0.0449624)	0.0930229(0.039692)
14	0.8	0.2	5.46893(0.269864)	0.326485(0.0540142)	0.0569272(0.0359603)
15	0.8	0.2	6.01652(0.261727)	0.362034(0.0564617)	0.0817849(0.0308793)
3	0.8	0.8	1.04665(0.0432815)	0.0207944(0.0118242)	0.0(0.0)
4	0.8	0.8	1.6081(0.0562091)	0.0762462(0.0216879)	0.0(0.0)
5	0.8	0.8	2.39829(0.0549469)	0.0733694(0.0223444)	0.0(0.0)
6	0.8	0.8	3.02212(0.0567535)	0.0872323(0.0239395)	0.0219433(0.0218333)
7	0.8	0.8	3.80538(0.0567323)	0.15367(0.0318464)	0.0616014(0.02526)
8	0.8	0.8	4.47773(0.0513303)	0.180875(0.0390725)	0.0680312(0.0468534)
9	0.8	0.8	5.1986(0.0541365)	0.162188(0.0372266)	0.0712856(0.03593)
10	0.8	0.8	5.8363(0.0538518)	0.193968(0.0404172)	0.0648062(0.0241476)
11	0.8	0.8	6.63342(0.0482771)	0.242613(0.0448573)	0.118454(0.0395468)
12	0.8	0.8	7.30577(0.0549469)	0.222182(0.0436092)	0.164913(0.052674)

**Figure 2, page 2**
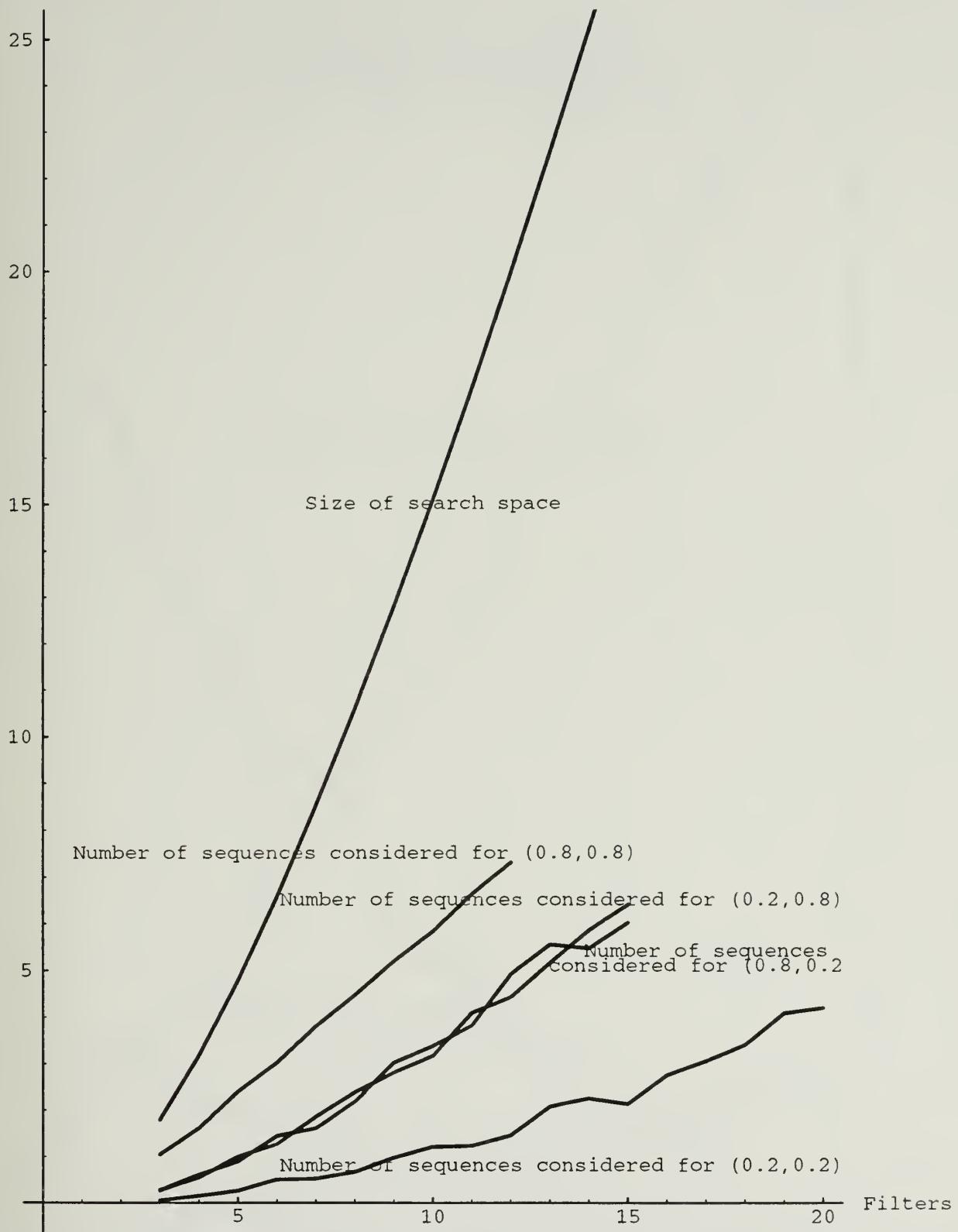
Figure 3

Mean Logarithm
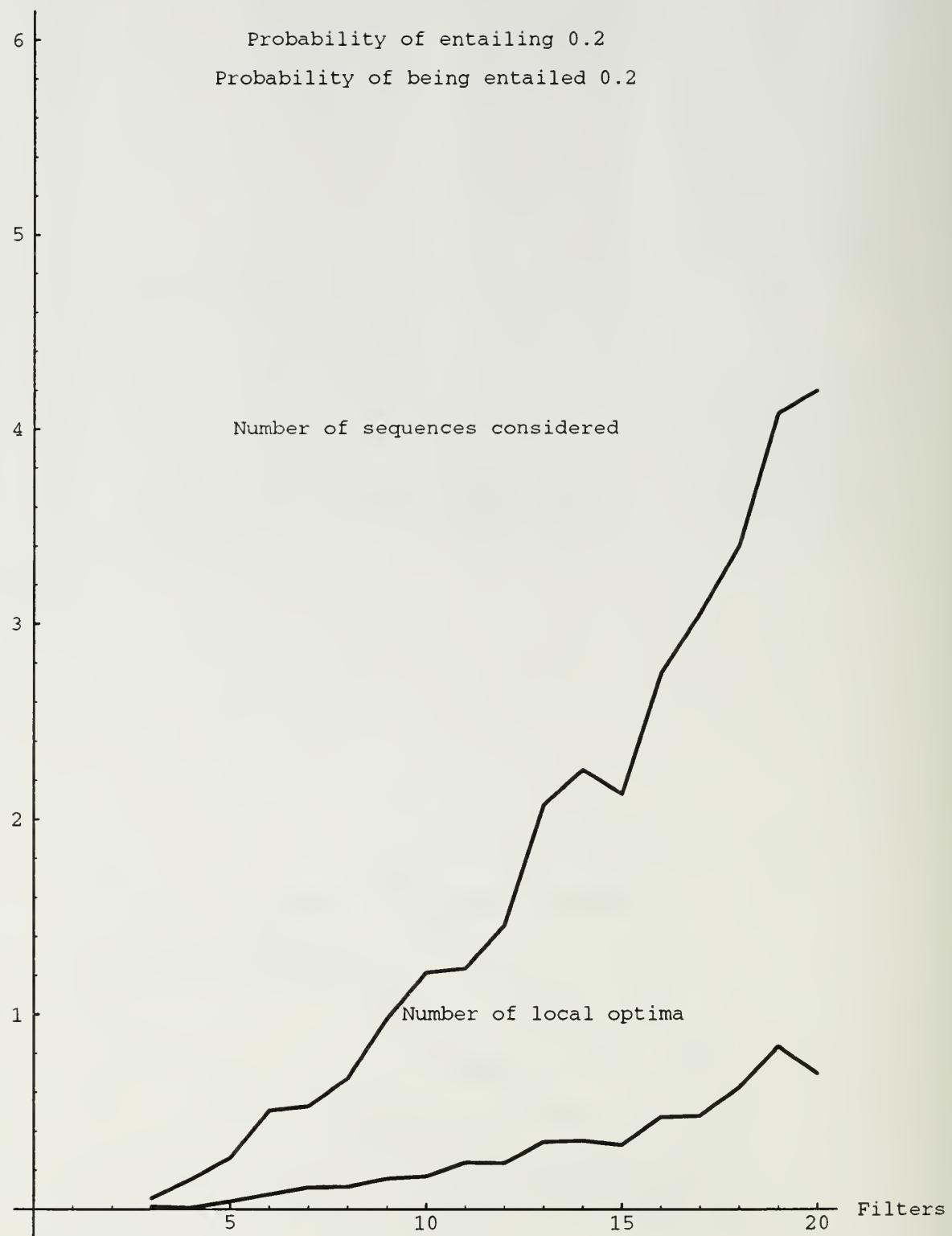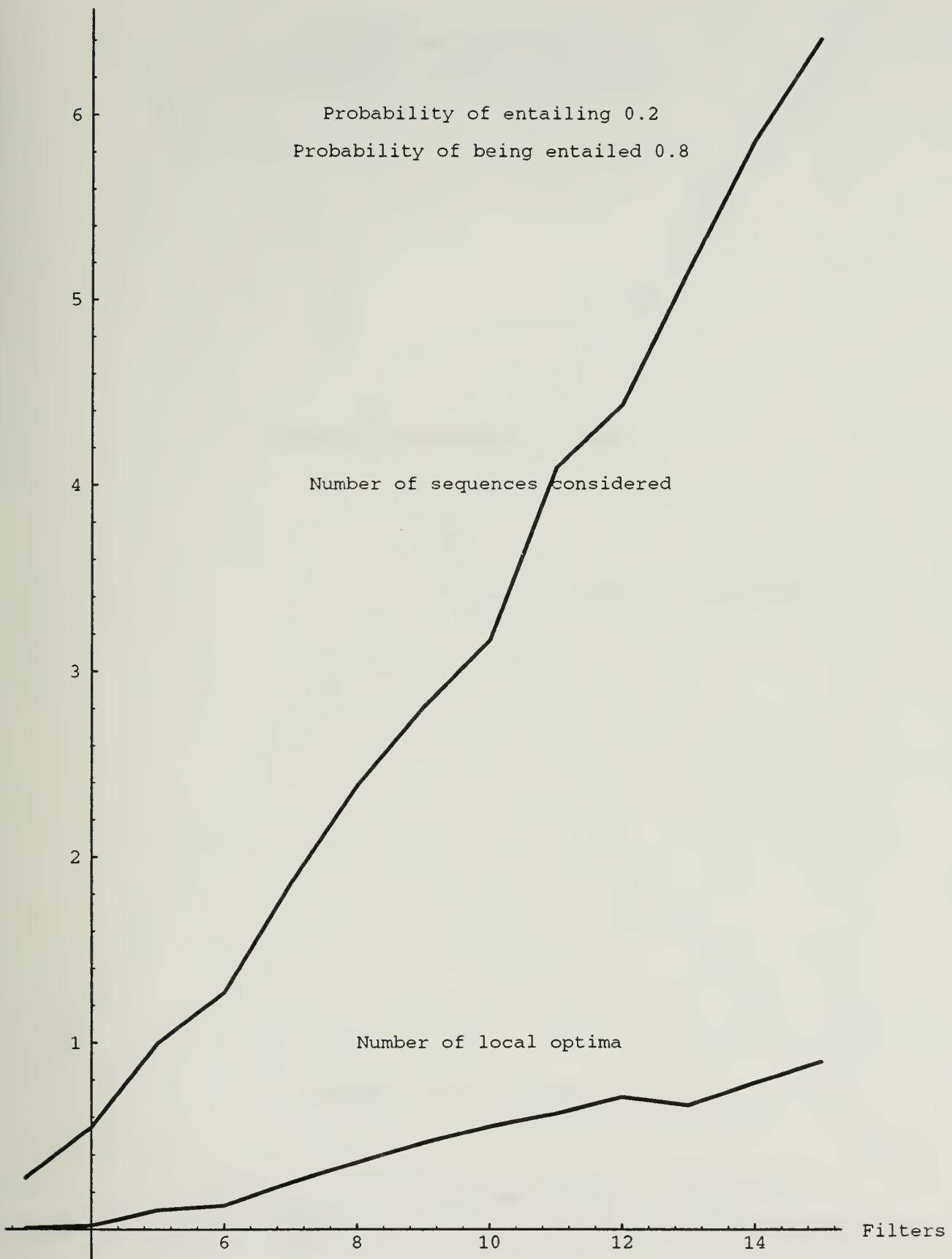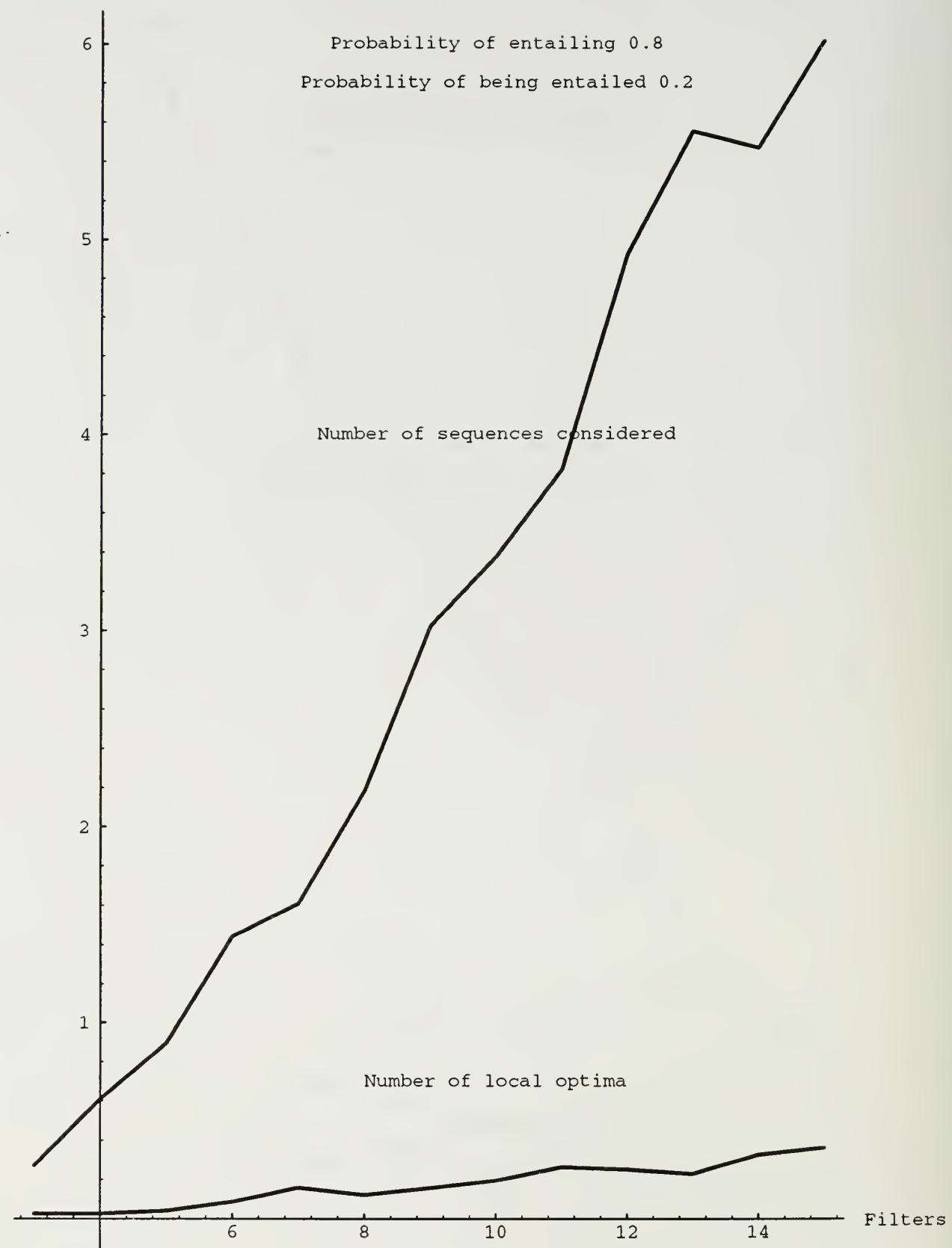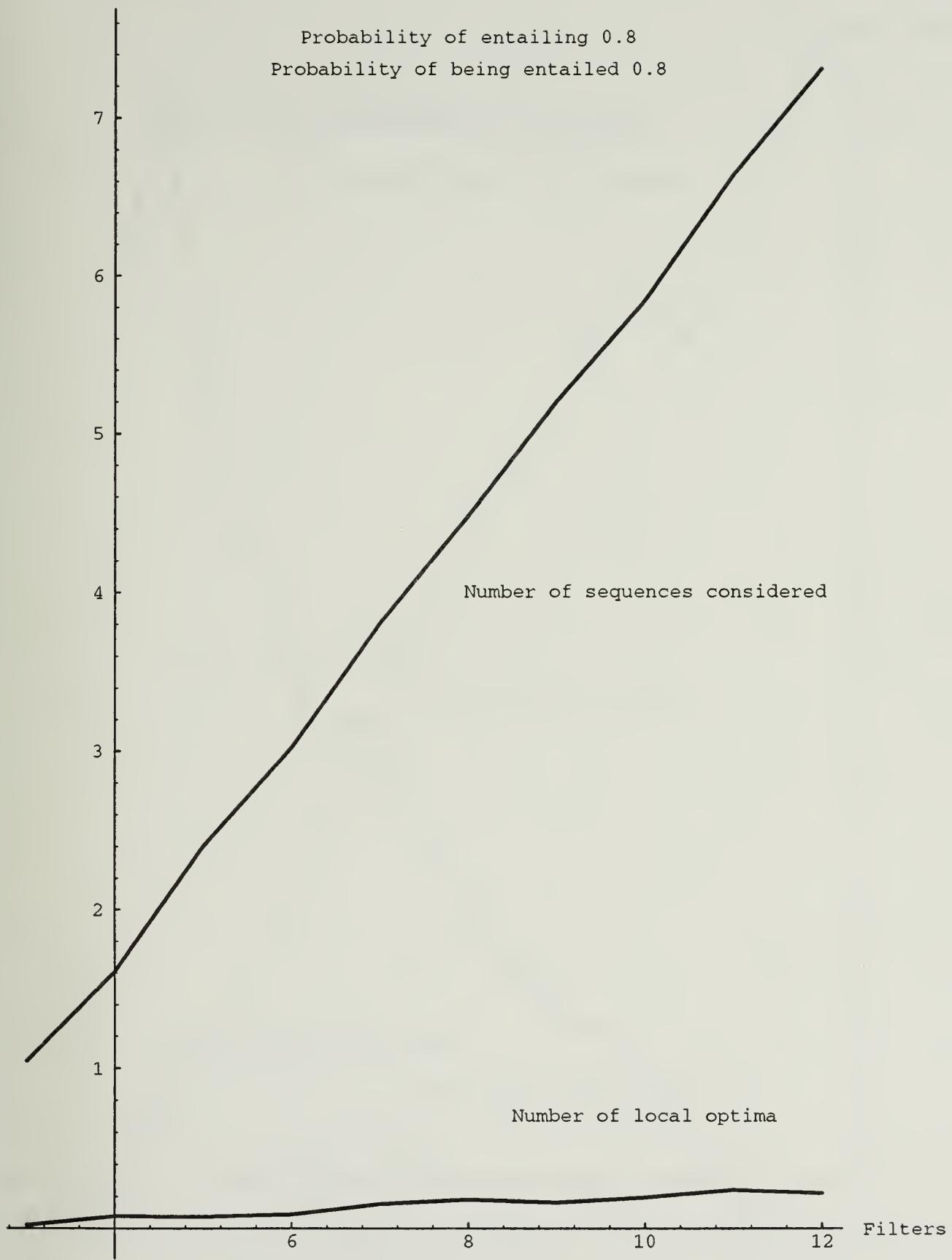
Probability of entailing 0.2
Probability of being entailed 0.2

Number of sequences considered

Number of local optima

Filters

Figure 4

Figure 5

Figure 6

Figure 7

Figure 8

Mean Logarithm

Probability of entailing 0.2
Probability of being entailed 0.8

Number of local optima

Cost ratio greedy / true

Filters
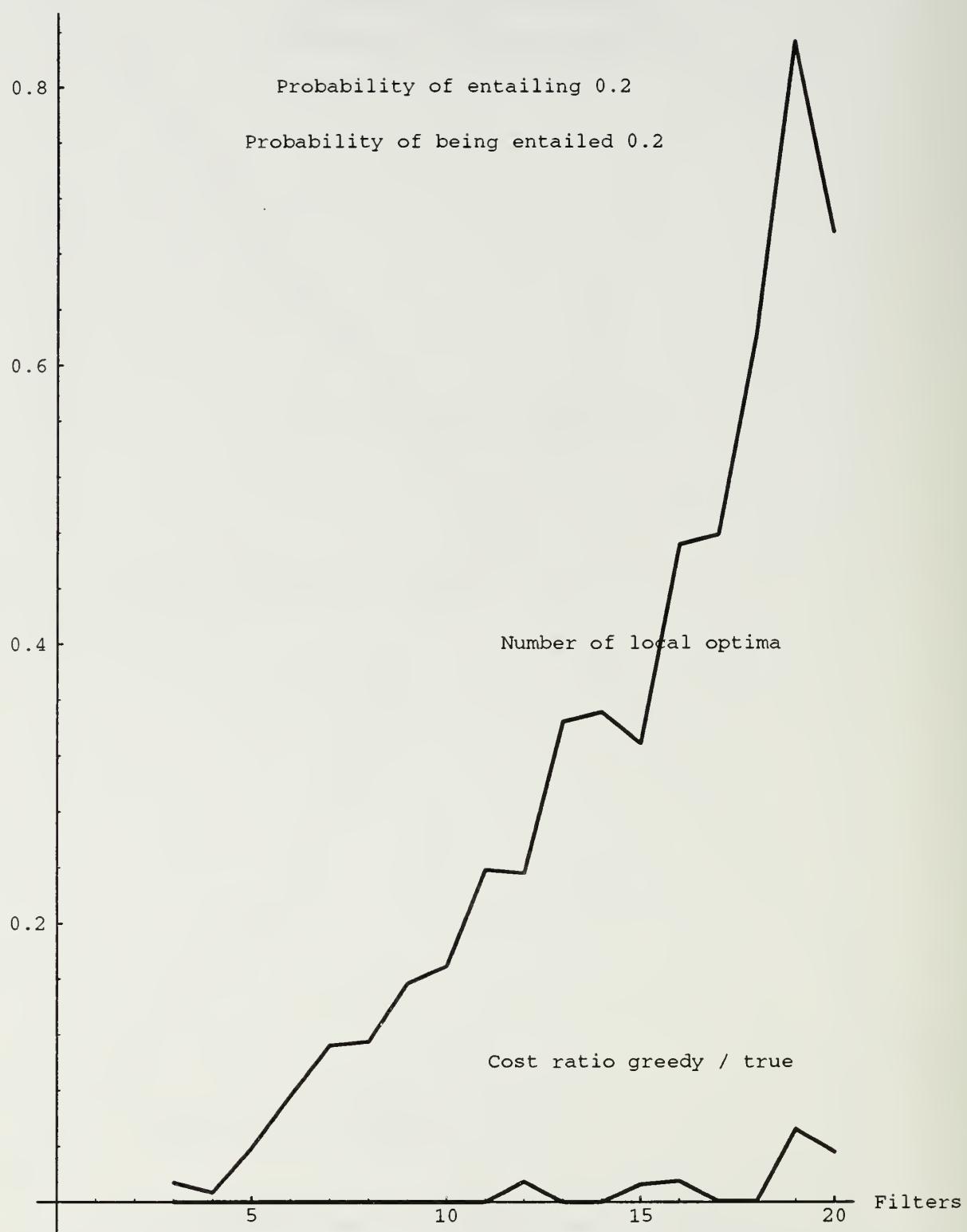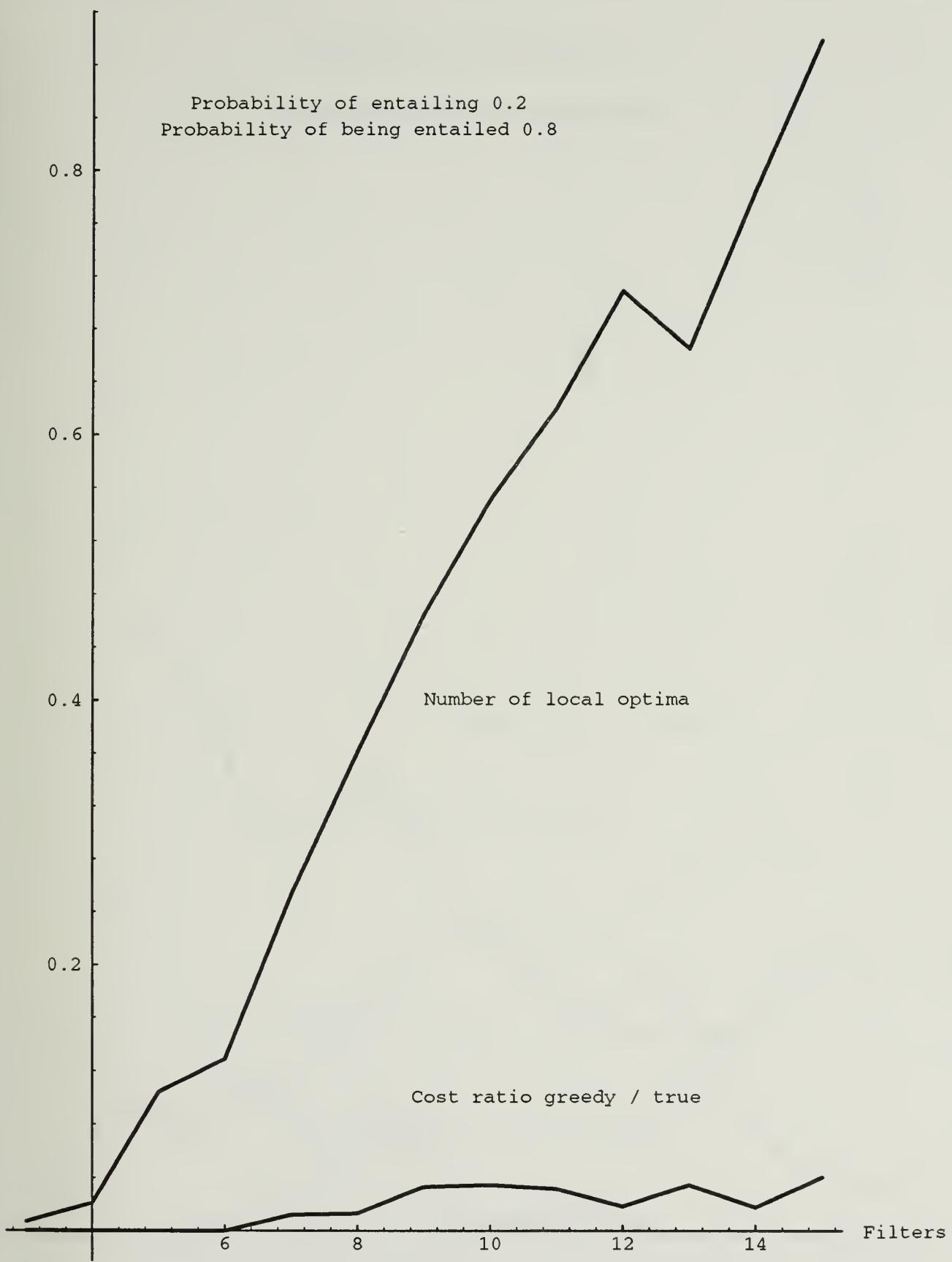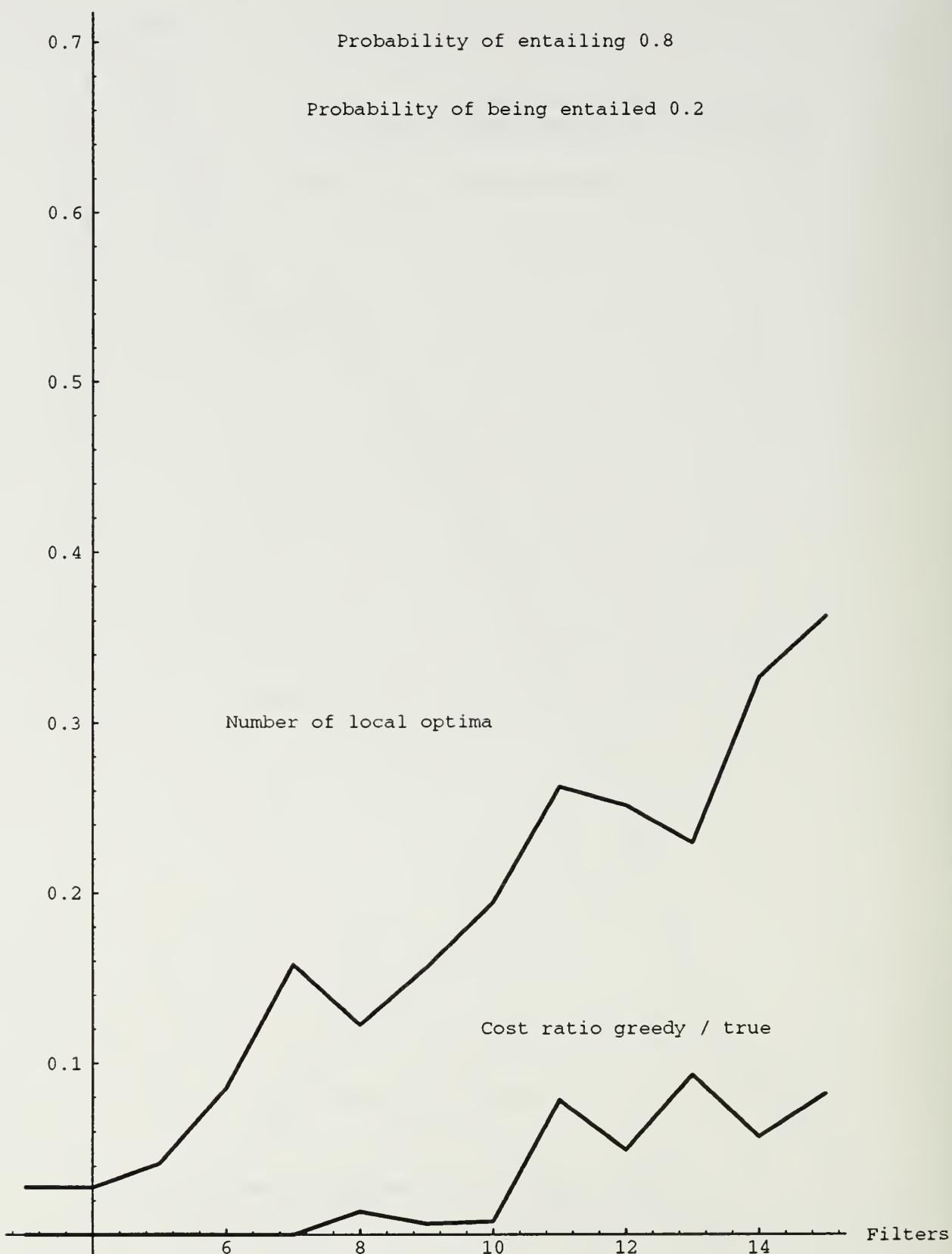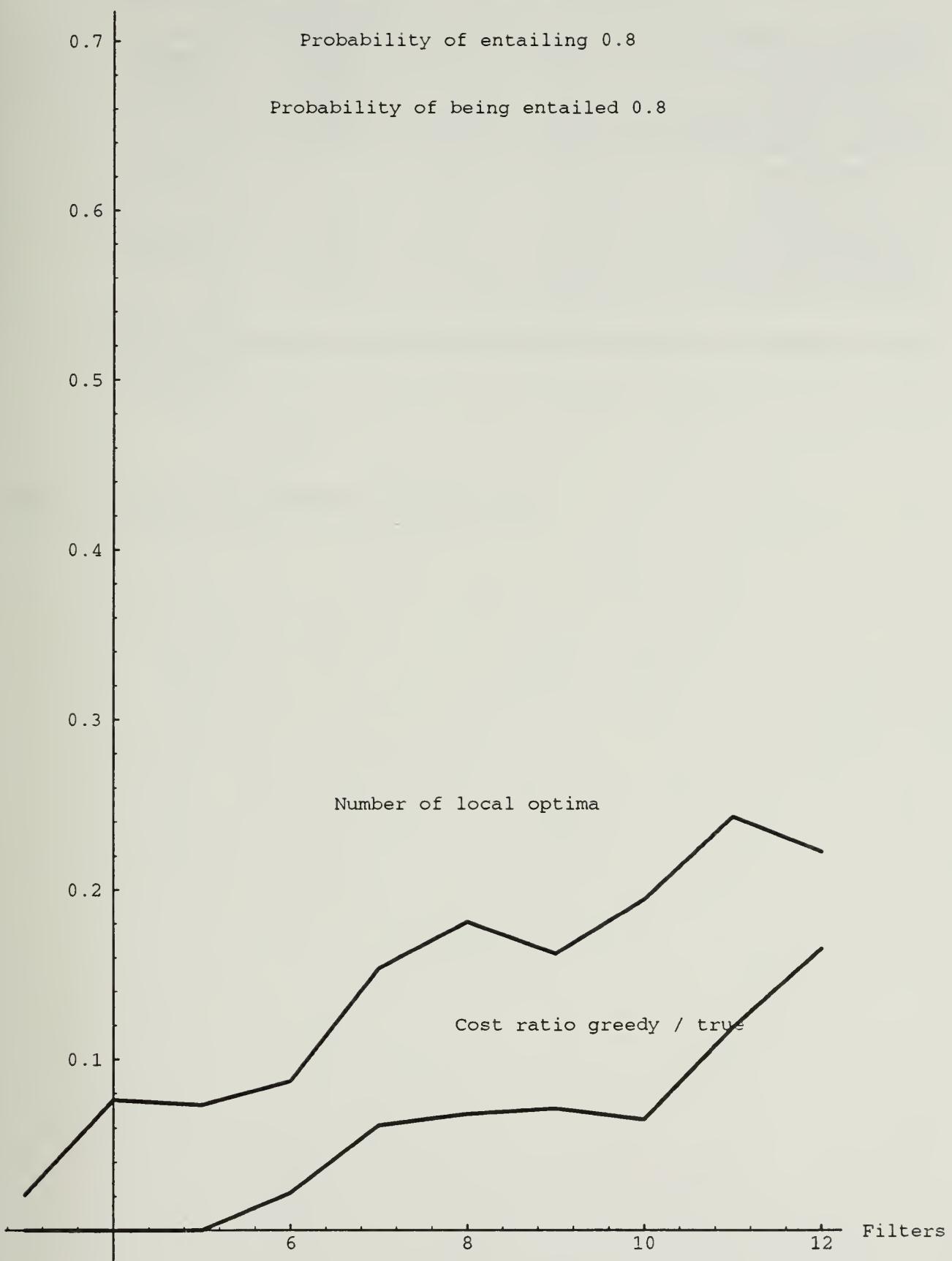
Figure 9

Figure]0

Figure 11

Class of logical equivalences	Optimal independent of costs and probabilities?	Satisfied by a greedy (nonbacktracking) algorithm?
Conjunctive commutivity	no	yes
Conjunctive elimination of entailed filters	no	no
Disjunctive commutivity	no	yes
Disjunctive elimination of entailed filters	no	no
Various redundancy eliminations	yes	yes
Distributivity factoring	yes	yes
DeMorgan's Laws inward	yes	no

**Figure 12: Summary of the optimality status of the standard boolean manipulations**
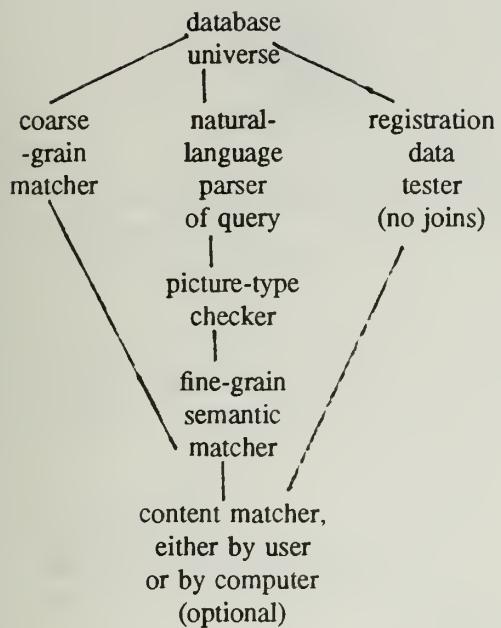
Figure 13: Dependencies between filters in the MARIE system

```
        coarse                                natural-
        -grain                                language
        matcher                                parser
     (1 processor)                          (1 processor)
           |
           |
     registration-
      data tester
     (1 processor)
           |
           |
     picture-type
        matcher
    (all processors)
           |
           |
      fine-grain
        matcher
    (all processors)
```
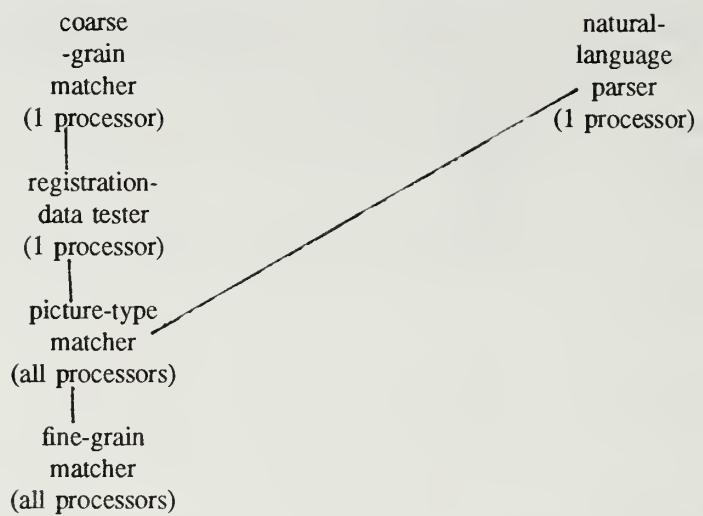
Figure 14: Optimal execution plan for the MARIE system, following the analysis in the text

Distribution List

Defense Technical Information Center
Cameron Station
Alexandria, VA 22314   1

Library, Code 52
Naval Postgraduate School
Monterey, CA 93943   1

Research Office
Code 08
Naval Postgraduate School
Monterey, CA 93943   1

Dr.Neil C. Rowe, Code CSRp
Naval Postgraduate School
Computer Science Department
Monterey, CA 93943-5118   50

Mr. Russell Davis
HQ, USACDEC
Office of Naval Research
Attention: ATEC-1M   1
Fort Ord, CA 93941

Ralph Wachter, Code 333
Computer Science
Office of Naval Research
Ballston Tower One
800 North Quincy St.
Arlington, VA 22217-5660   1